



(51) International Patent Classification:

G06N 3/045 (2023.01) G06N 3/09 (2023.01)
G06N 3/084 (2023.01)

(21) International Application Number:

PCT/EP2024/082601

(22) International Filing Date:

15 November 2024 (15.11.2024)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

63/600,476 17 November 2023 (17.11.2023) US

(71) Applicant: **DEEPMIND TECHNOLOGIES LIMITED**
[GB/GB]; 5 New Street Square, London EC4A 3TW (GB).

(72) Inventors: **BICA, Ioana**; 6 Pancras Square, London N1C 4AG (GB). **ILIC, Anastasija**; 6 Pancras Square, London N1C 4AG (GB). **BAUER, Matthias Stephan**; 6 Pancras Square, London N1C 4AG (GB). **ERDOGAN, Goker**; 6 Pancras Square, London N1C 4AG (GB). **BOSNJAK, Matko**; 6 Pancras Square, London N1C 4AG (GB). **KAPLANIS, Christos**; 6 Pancras Square, London N1C 4AG (GB). **GRITSENKO, Alexey Alexeevich**; Claude Debussylaan 34, 1082 MD Amsterdam (NL). **MINDER-**

ER, Matthias Johannes Lorenz; Brandschenkestrasse 110, 8002 Zürich (CH). **BLUNDELL, Charles**; 6 Pancras Square, London N1C 4AG (GB). **PASCANU, Razvan**; 6 Pancras Square, London N1C 4AG (GB). **MITROVIC, Jovana**; 6 Pancras Square, London N1C 4AG (GB).

(74) Agent: **FISH & RICHARDSON P.C.**; HighLight Business Towers Mies-van-der-Rohe-Str. 8, 80807 München (DE).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CV, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, MG, MK, MN, MU, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, CV, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SC, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ,

(54) Title: TRAINING IMAGE PROCESSING NEURAL NETWORKS USING CROSS-MODAL ALIGNMENT

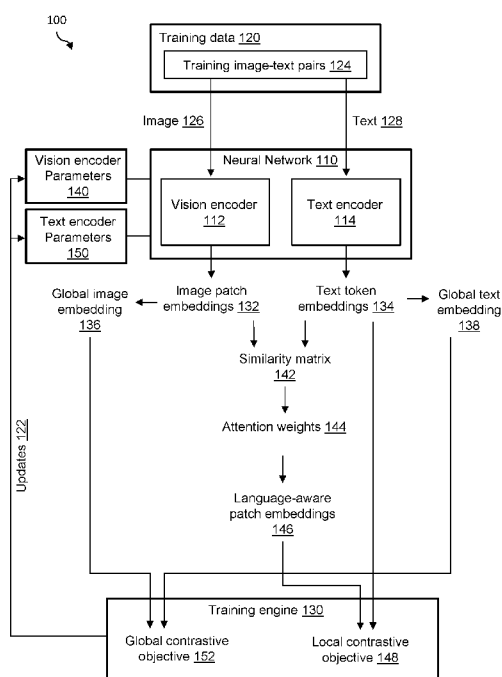


Figure 1

(57) Abstract: A computer-implemented method of training a neural network system comprising a visual encoder neural network and a text encoder neural network is provided. The method comprises obtaining a plurality of training data items (each training data item comprising an image and associated text defining a sequence of text tokens) and at each of a plurality of training steps processing at least one of the training data items by: processing pixels of the image in the training data item using the visual encoder neural network to generate a set of patch embeddings for the image; processing the sequence of text tokens using the text encoder neural network to generate a sequence of token embeddings, processing the set of patch embeddings and the sequence of token embeddings to generate a set of language-aware patch embeddings (based on similarities between patch embeddings and token embeddings), and training at least the visual encoder neural network by back-propagating gradients of a contrastive objective function evaluated over the language-aware patch embeddings and the sequence of token embeddings.

DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT,
LU, LV, MC, ME, MK, MT, NL, NO, PL, PT, RO, RS, SE,
SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN,
GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

- *with international search report (Art. 21(3))*
- *in black and white; the international application as filed
contained color or greyscale and is available for download
from PATENTSCOPE*

TRAINING IMAGE PROCESSING NEURAL NETWORKS USING CROSS-MODAL ALIGNMENT

CROSS-REFERENCE TO RELATED APPLICATION

[01] This application claims priority to U.S. Provisional Application No. 63/600,476, filed on November 17, 2023. The disclosure of the prior application is considered part of and is incorporated by reference in the disclosure of this application.

BACKGROUND

[02] This specification relates to processing data using machine learning models.

[03] Neural networks are machine learning models that employ one or more layers of nonlinear units to predict an output for a received input. Some neural networks include one or more hidden layers in addition to an output layer. The output of each hidden layer is used as input to the next layer in the network, i.e., the next hidden layer or the output layer. Each layer of the network generates an output from a received input in accordance with current values of a respective set of parameters.

SUMMARY

[04] This specification describes a system and method, implemented as computer programs on one or more computers in one or more locations, for training a visual encoder neural network. The visual encoder neural network learns representations of images using cross-modal alignment, in particular through contrastive learning. Some implementations also train a text encoder neural network. Implementations of the system can quickly learn good representations.

[05] In a first aspect there is described a computer-implemented method of training a neural network system comprising a visual encoder neural network and a text encoder neural network, each having a plurality of (trainable) parameters such as weights. The method can be used to train the neural network(s) from scratch (from random initialization), or to fine tune the neural network(s). The method comprises obtaining a dataset of training data comprising a plurality of training data items. Each training data item comprises an image and associated text. The associated text defines a sequence of text tokens from a vocabulary of tokens. The method further comprises, at each of a plurality of training steps processing at least one of the training data items by: processing pixels of the image in the training data item using the visual encoder neural network to generate a set of patch embeddings for

the image (each patch embedding representing values of the pixels defining the image content of a corresponding patch of the image), processing the sequence of text tokens using the text encoder neural network to generate a sequence of token embeddings representing the sequence of text tokens, and processing the set of patch embeddings and the sequence of token embeddings to generate a set of language-aware patch embeddings. Each language-aware patch embedding is determined from the set of patch embeddings based on similarities between patch embeddings of the set of patch embeddings and token embeddings of the sequence of token embeddings. The method further comprises training at least the visual encoder neural network by backpropagating gradients of a contrastive objective function evaluated over the language-aware patch embeddings and the sequence of token embeddings, to adjust the parameters of the visual encoder neural network.

[06] According to a second aspect there is provided a computer-implemented method of performing an image processing task. The method comprises receiving one or more patch embeddings and/or a global image embedding generated using a visual encoder neural network trained according to the first aspect, and using the one or more patch embeddings and/or the global image embedding to perform an image processing task. Performing the image processing task comprises: processing pixels of an image to generate the set of patch embeddings and/or global image embedding for the image, and using the set of patch embeddings and/or global image embedding to perform the image processing task. The image processing task comprises: processing the image to provide output data that identifies the presence or location of one or more objects in the image, or processing the image to provide output data that segments pixels of the image into regions that represent one or more objects in the image, or processing the image to provide output data that categorizes a content of the image into one or more of a plurality of categories, or processing the image to provide output data that predicts depth values for pixels of the image, or processing the image to provide output data that is used to control an action of a mechanical agent acting in a real-world environment to perform a task, or when the image comprises a moving image, the image processing task comprises: processing the image to provide output data that identifies the location of one or more actions represented in the video, or processing the image to provide output data that categorizes one or more actions represented in the video into one or more of a plurality of categories, or processing the image to provide output data that predicts one or more events that involve objects in the image, or when the method further comprises receiving a sequence of token embeddings from a trained text encoder neural network, processing the one or more patch embeddings and/or the global image embedding and the

sequence of token embeddings to perform an image processing task defined by a sequence of text tokens encoded by the sequence of token embeddings.

[07] According to a third aspect there is provided a computer-implemented method of performing an image processing task. The method comprises providing an image to a visual encoder neural network that has been trained by performing the respective operations of the method of the first aspect, processing pixels of the image using the visual encoder neural network to generate the set of patch embeddings or the global image embedding for the image, and processing the set of patch embeddings or the global image embedding for the image to perform the image processing task. The image processing task comprises: processing the image to provide output data that identifies the presence or location of one or more objects in the image, or processing the image to provide output data that segments pixels of the image into regions that represent one or more objects in the image, or processing the image to provide output data that categorizes a content of the image into one or more of a plurality of categories, or processing the image to provide output data that predicts depth values for pixels of the image, or processing the image to provide output data that is used to control an action of a mechanical agent acting in a real-world environment to perform a task, or when the image comprises a moving image, the image processing task comprises: processing the image to provide output data that identifies the location of one or more actions represented in the video, or processing the image to provide output data that categorizes one or more actions represented in the video into one or more of a plurality of categories, or processing the image to provide output data that predicts one or more events that involve objects in the image, or when the method further comprises receiving a sequence of token embeddings from a trained text encoder neural network, processing the one or more patch embeddings and/or the global image embedding and the sequence of token embeddings to perform an image processing task defined by a sequence of text tokens encoded by the sequence of token embeddings.

[08] According to a fourth aspect there is provided a computer-implemented method of processing, using a neural network system comprising a visual encoder neural network and a text encoder neural network, an image and associated text to generate a set of language-aware patch embeddings. The method comprises obtaining the image and the associated text (wherein the associated text defines a sequence of text tokens from a vocabulary of tokens), processing pixels of the image using the visual encoder neural network to generate a set of patch embeddings for the image (each patch embedding representing values of the pixels defining the image content of a corresponding patch of the image), processing the sequence

of text tokens using the text encoder neural network to generate a sequence of token embeddings representing the sequence of text tokens, and processing the set of patch embeddings and the sequence of token embeddings to generate the set of language-aware patch embeddings. Each language-aware patch embedding is determined from the set of patch embeddings based on similarities between patch embeddings of the set of patch embeddings and token embeddings of the sequence of token embeddings.

[009] According to a fifth aspect there is provided a system comprising one or more computers, and one or more storage devices communicatively coupled to the one or more computers. The storage devices store instructions that, when executed by the one or more computers, cause the one or more computers to perform the operations of the first, second, third, or fourth aspect.

[010] According to a sixth aspect there is provided one or more non-transitory computer storage media storing instructions that when executed by one or more computers perform the operations of the first, second, third, or fourth aspect.

[011] According to a seventh aspect there is provided a visual encoder neural network comprising a plurality of (trainable/trained) parameters and configured to process pixels of an image to generate a set of patch embeddings for the image. Each patch embedding represents values of the pixels defining the image content of a corresponding patch of the image. The patch embeddings and corresponding text token embeddings represent a textual description of the image that have a similarity relationship, according to a similarity metric, such that patch embeddings corresponding to image regions described by a portion of the textual description have a higher similarity to the text token embedding representing said portion of the textual description than to text token embeddings representing other portions of the textual description.

[012] According to an eighth aspect there is provided a method of performing an image processing task. The method comprises receiving one or more patch embeddings for an image, each patch embedding representing values of pixels defining the image content of a corresponding patch of the image, wherein the patch embeddings and corresponding text token embeddings representing a textual description of the image have a similarity relationship, according to a similarity metric, such that patch embeddings corresponding to image regions described by a portion of the textual description have a higher similarity to the text token embedding representing said portion of the textual description than to text token embeddings representing other portions of the textual description; and using the one or more patch embeddings to perform an image processing task.

[013] According to a ninth aspect there is provided a set of patch embeddings for an image. Each patch embedding represents values of the pixels defining the image content of a corresponding patch of the image. The patch embeddings and corresponding text token embeddings representing a textual description of the image have a similarity relationship, according to a similarity metric, such that patch embeddings corresponding to image regions described by a portion of the textual description have a higher similarity to the text token embedding representing said portion of the textual description than to text token embeddings representing other portions of the textual description.

[014] The subject matter described in this specification can be implemented in particular embodiments so as to realize one or more of the following advantages.

[015] Training a visual encoder neural network and a text encoder neural network using just global representations can result in poor performance on some downstream tasks such as semantic localization, object detection, semantic segmentation, counting, and understanding spatial relationships between objects or object attributes. Incorporating local losses between patch embeddings and text token embeddings can help to improve performance, but computing similarities between all patch embeddings and text token embeddings in a batch of text-image pairs can be computationally expensive. Further, using a softmax for computing attention weights implicitly assumes that there is a one-to-one mapping between each text token and image patch, which is often not the case. For example, the text token embedding for “dog” should correspond to all patch embeddings that correspond to a dog in the image. In addition, use of a softmax function can result in optimization difficulties due to inhibited gradient flow.

[016] Implementations of the described techniques address all of these problems. The described techniques can result in both faster training and improved performance on many types of fine-grained and coarse-grained vision tasks, as confirmed in experiments described below in more detail. Further, unlike other methods which rely on pre-trained vision encoders and/or pre-trained text encoders, the system described in this specification can be used for training both vision encoders and pre-trained text encoders from scratch. The described techniques are also computationally efficient, scaling to large training datasets. Furthermore, it is also advantageous that the system described in this specification can be combined with standard vision encoder architectures. That is, in broad terms, the proposed system enables a vision encoder neural network having a standard vision transformer architecture but improved performance because of a novel training objective that encourages fine-grained understanding.

[017] In certain implementations, a similarity matrix representing the similarities between patch embeddings of the set of patch embeddings and token embeddings of the sequence of token embeddings may be determined (each element of the similarity matrix representing a similarity between one of the patch embeddings and one of the token embeddings) and the set of language-aware patch embeddings may be generated from the set of patch embeddings based on the similarity matrix (without the use of a softmax function). As an example, a set of attention weights for combining the patch embeddings may be generated from the elements of the similarity matrix (rather by using a softmax function). This can avoid the above-mentioned problems associated with using a softmax function to compute the attention weights for combining the patch embeddings.

[018] In certain implementations, the similarity matrix may be sparsified (e.g. processed to set a subset of the elements of the similarity matrix to zero), and the set of language-aware patch embeddings are generated from the set of patch embeddings based on the sparsified similarity matrix processing. This sparsification facilitates learning by encouraging that only the most relevant image patches contribute to each language-aware patch embedding.

[019] In certain implementations, a batch of the training data items may be processed at each training step, and a global image embedding and a global text embedding may be generated for each training data item. A global contrastive objective for the batch of training data items may be evaluated based on the global image embeddings and the global text embeddings. The visual encoder neural network and text encoder neural network can then be trained (jointly) by backpropagating gradients of the global contrastive objective function, e.g. to maximize the similarity to corresponding global text and image embeddings while minimizing the similarity to other global text and image embeddings in the batch. As an example, an overall objective function for training the visual encoder neural network and text encoder neural network may comprise a weighted combination of the (local) contrastive objective function and the global contrastive objective function. Training the visual encoder neural network using both local and global contrastive objective functions enables the visual encoder neural network to learn representation that that encode both coarse-grained/global and fine-grained/local information.

[020] The details of one or more embodiments of the subject matter of this specification are set forth in the accompanying drawings and the description below. Other features, aspects, and advantages of the subject matter will become apparent from the description, the drawings, and the claims.

BRIEF DESCRIPTION OF THE DRAWINGS

- [021] FIG. 1 shows an example system for training a visual encoder neural network.
- [022] FIG. 2 is a flow diagram of an example process for training a visual encoder neural network.
- [023] FIG. 3 illustrates an example training iteration.
- [024] FIGS. 4 to 10 show experimental results generated by the described techniques.
- [025] Like reference numbers and designations in the various drawings indicate like elements.

DETAILED DESCRIPTION

- [026] FIG. 1 shows a system 100 implemented as computer programs on one or more computers in one or more locations, for training a visual encoder neural network 112.
- [027] The system 100 comprises a neural network 110, training data 120 and a training engine 130. The neural network 110 comprises the visual encoder neural network 112 defined by a plurality of (trainable) parameters 140, and a text encoder neural network 114 defined by a respective plurality of (trainable) parameters 150. The system 100 is configured to process image-text pairs to iteratively update (i.e. train) at least the parameters 140 of the visual encoder neural network 112 (e.g. in a self-supervised manner). More specially, in each one of a plurality of iterations (or “training steps”), the system 100 generates a corresponding parameter update 122 for at least the parameters 140 of the visual encoder neural network 112. In some implementations, the update 122 may also comprise a corresponding update for the parameters 150 of the text encoder neural network 114.
- [028] The update 122 is generated by the training engine 130 based on output of the neural network 110 generated for input data selected from the training data 120. The training data 120 is a dataset of training data comprising a plurality of training data items 124. Each training data item 124 comprises an image, which may be still or moving (i.e. the image may be an image of a sequence of images defining a video), in 2D or in 3D, and associated text. The associated text (or “caption”) defines a sequence of text tokens from a vocabulary of tokens. The tokens can each represent, e.g., words, wordpieces or characters in a natural or computer language.
- [029] There is a wide range of datasets available that can be used for training the system 100, e.g. ALIGN, JFT, LTIP, and the like.
- [030] In general, the visual encoder neural network 112 is configured to receive an input comprising pixels of a still or moving image 126 (e.g. an image of a training data item 124), and to process the input, in accordance with the current values of trainable/trained parameters

140 of the visual encoder neural network 112, to generate as an output a set of patch embeddings 132 for the image, and optionally from them the global image embedding 136. Each patch embedding can represent values of the pixels defining the image content of a corresponding patch (region) of the image 126.

[031] More specifically, denoting the image 126 as $\mathbf{x}^v$, the corresponding patches of the image may be denoted as $(\mathbf{x}_1^v, \mathbf{x}_2^v, \mathbf{x}_3^v, \dots, \mathbf{x}_P^v)$, where P is the number of patches. The visual encoder neural network 112 may model an image encoder function $f_v(\cdot)$ to generate the set of patch embeddings 132, denoted $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \dots, \mathbf{v}_P)$, according to $\mathbf{v}_p = f_v(\mathbf{x}_p^v)$, where p indicates a respective image patch. In some implementations, the visual encoder neural network 112 may generate the set of patch embeddings 132 using the image encoder function $f_v(\cdot)$ and a linear transformation $g_v(\cdot)$ (also referred to as “linear adapter”). As an example, the visual encoder neural network 112 may generate the set of patch embeddings 132 according to $\mathbf{v}_p = g_v(f_v(\mathbf{x}_p^v)) \in \mathbb{R}^d$, where d is the number of elements in $\mathbf{v}_p$. The linear adapter g_v may project (or map) the output of the function f_v modelled by the visual encoder neural network 112 into an embedding space that is shared between said patch embeddings and text embeddings generated by the text encoder neural network 114, as described below in more detail. As noted above, in some implementations, the use of the linear adapter g_v may not be necessary.

[032] In some implementations, the system 100 may process a (mini-)batch of the training data items 124, e.g. at each of the training steps. The batch of the training data items 124 may be denoted $\mathcal{B} = \{(\mathbf{x}_1^v, \mathbf{x}_1^t), (\mathbf{x}_2^v, \mathbf{x}_2^t), \dots, (\mathbf{x}_B^v, \mathbf{x}_B^t)\}$, where B is the number of training data items, i.e. image-text pairs, in the batch. In this case, the visual encoder neural network 112 may be configured to generate a respective set of patch embeddings 132 for each training data item in the batch, e.g. $(\mathbf{v}_{i,1}, \mathbf{v}_{i,2}, \mathbf{v}_{i,3}, \dots, \mathbf{v}_{i,P})$ may denote the set of patch embeddings 132 for the i-th training data item in the batch.

[033] In some implementations, the visual encoder neural network 112 may be further configured determine a global image embedding 136 for the image 126 by combining the set of patch embeddings 132 for the image 126. More specifically, when the system 100 processes a batch of the training data items 124, the visual encoder neural network 112 may be configured to generate a respective global image embedding for each training data item in the batch.

[034] Many possibilities exist for combining the set of patch embeddings 132 to generate the global image embedding 136 for the image 126. As an example, combining the set of

patch embeddings 132 for the image 126 can involve average pooling the set of patch embeddings.

[035] In some implementations the combined set of patch embeddings may be further processed by a non-linear layer to determine the global image embedding. The combined set of patch embeddings can be processed by the non-linear layer or each patch embedding can be processed by the non-linear layer and then the processed patch embeddings can be combined. This can facilitate retaining information at different levels of granularity. As an example, the visual encoder neural network 112 may generate the global image embedding $\bar{v}_i$ for the i-th

training data item in the batch according to $\bar{v}_i = g_v \left(h_v \left(\text{avg_pool} \left(\{f_v(x_{i,p}^v)\}_{p=1}^P \right) \right) \right)$, where $\text{avg_pool}(\cdot)$ indicates an average pooling over an input “.” and $h_v(\cdot)$ is a single non-linear layer that facilitates retaining information at a different levels of granularity. As described below in more detail, the global image embeddings may be used in the evaluation of a global contrastive objective to enable learning of representations that also encode global information.

[036] In general, the text encoder neural network 114 is configured to receive an input comprising text 128 (e.g. text of a training data item 124, i.e. text associated with an image of a training data item 124) defining a sequence of text tokens from a vocabulary of tokens, and to process the input, in accordance with the current values of the trainable/trained parameters 150 of the text encoder neural network 114, to generate as an output the sequence of token embeddings 134 for the text 128, and optionally from them the global text embedding 138. The sequence of token embeddings 134 may represent the sequence of text tokens.

[037] More specifically, denoting the text 128 as $\mathbf{x}^t$, the corresponding tokens may be denoted as $(\mathbf{x}_1^t, \mathbf{x}_2^t, \mathbf{x}_3^t, \dots, \mathbf{x}_L^t)$, where L is the number of tokens in the text $\mathbf{x}^t$. The text encoder neural network 114 may model a text encoder function $f_t(\cdot)$ to generate the sequence of token embeddings 134, denoted $(\mathbf{t}_1, \mathbf{t}_2, \mathbf{t}_3, \dots, \mathbf{t}_L)$, according to $\mathbf{t}_l = f_t(\mathbf{x}_l^t)$, where l indicates a respective token. In some implementations, the text encoder neural network 114 may generate the sequence of token embeddings 134 using the text encoder function $f_t(\cdot)$ and a linear adapter $g_t(\cdot)$. As an example, the text encoder neural network 114 may generate the sequence of token embeddings 134 according to $\mathbf{t}_l = g_t(f_t(\mathbf{x}_l^t))$. In some implementations, the linear adapter g_t may project (or map) the output of the function f_t modelled by the text encoder neural network 114 into an embedding space that is shared between token embeddings 134 and the patch embeddings 132 (as noted above, in other implementations, the use of the linear adapter g_t may not be necessary). Thus, if necessary the patch embeddings 132 and the token embeddings

134 can be mapped to a shared embedding space so that they each have the same dimension, e.g. by applying a linear adapter to one or both of these embeddings.

[038] As noted above, in some implementations, the system 100 may process a batch $\mathcal{B} = \{(\mathbf{x}_1^v, \mathbf{x}_1^t), (\mathbf{x}_2^v, \mathbf{x}_2^t), \dots, (\mathbf{x}_B^v, \mathbf{x}_B^t)\}$ of the training data items 124 at each of the training steps. In this case, the text encoder neural network 114 may be configured to generate a respective sequence of token embeddings 134 for each training data item in the batch, e.g. $(\mathbf{t}_{i,1}, \mathbf{t}_{i,2}, \mathbf{t}_{i,3}, \dots, \mathbf{t}_{i,L_i})$ may denote the sequence of token embeddings 134 for the i -th training data item in the batch, where L_i is the number of tokens in the text of the i -th data item in the batch.

[039] In some implementations, the text encoder neural network 114 may be further configured determine a global text embedding 138 for the text 128 by combining the sequence of token embeddings 134 for the text 128. More specifically, when the system 100 processes a batch of the training data items 124, the text encoder neural network 114 may be configured to generate a respective global text embedding for each training data item in the batch.

[040] Many possibilities exist for combining the sequence of token embeddings 134 to generate the global text embedding 138 for the text 128. As an example, combining the sequence of token embeddings 134 for the text 128 can involve average pooling the sequence of token embeddings. More specifically, the text encoder neural network 114 may generate the global text embedding $\bar{\mathbf{t}}_i$ for the i -th training data item in the batch according to $\bar{\mathbf{t}}_i = g_t \left(\text{avg_pool} \left(\{f_t(\mathbf{x}_{i,l}^t)\}_{l=1}^{L_i} \right) \right)$.

[041] The system 100 is further configured to process the set of patch embeddings 132 and the sequence of token embeddings 134 to generate a set of language-aware (or “language-grouped”) patch embeddings 146. Each language-aware patch embedding may be determined from the set of patch embeddings based on similarities, i.e. values of a metric of similarity, between patch embeddings of the set of patch embeddings and token embeddings of the sequence of token embeddings.

[042] In some implementations, the number of language-aware patch embeddings may match a length of the sequence of token embeddings, so that there can be a 1:1 correspondence between the language-aware patch embeddings and the token embeddings. For every token embedding, the system 100 may generate a corresponding language-aware patch embedding by combining embeddings of patches that encode that token in the visual domain. This is motivated by an observation by the inventors that often multiple image patches correspond to one word in an associated caption.

[043] More specifically, the system 100 may be configured to determine, for each token embedding (token), one language-aware patch embedding by combining the patch embeddings according to attention weights 144 (or “alignment weights”) for the token embedding and for the respective patch embeddings, where the attention weights 144 are computed based on a similarity between token embeddings and patch embeddings of a corresponding image-text pair. As an example, each language-aware patch embedding may be generated by combining the patch embeddings as a weighted average, weighted by the respective attention weights.

[044] Thus, to generate the set of language-aware patch embeddings 146, the system 100 may be configured determine, for each patch embedding of the set of patch embeddings 132, an attention weight for each token embedding. As noted above, determining the attention weight for each token embedding can involve, for each patch embedding and for each token embedding of the sequence of token embeddings 134, determining the attention weight for the patch embedding and for the token embedding from a metric of similarity between the patch embedding and the token embedding.

[045] In some implementations, generating the set of language-aware patch embeddings comprises determining a similarity matrix 142 representing the similarities between patch embeddings of the set of patch embeddings 132 and token embeddings of the sequence of token embeddings 134. Each element of the similarity matrix can represent a similarity, e.g. a metric of similarity, between one (of each) of the patch embeddings 132 and one (of each) of the token embeddings 134. The system 100 may be configured to generate the set of language-aware patch embeddings 146 from the set of patch embeddings 132 based on the similarity matrix 142.

[046] Generating the set of language-aware patch embeddings 146 from the set of patch embeddings 132 based on the similarity matrix 142 can involve generating each language-aware patch embedding by, for each patch embedding, determining a set of attention weights, one attention weight for each of the token embeddings, from the elements of the similarity matrix for that patch embedding and for each of the token embeddings. Some of the attention weights may be zero. The system 100 can determine a language-aware patch embedding, for each token embedding, by combining each of the patch embeddings according to the respective weight for that patch embedding and for the token embedding (i.e. for the corresponding token).

[047] The similarity matrix 142 may be determined using any suitable metric of similarity. As an example, each element of the similarity matrix may be determined based on an inner

product of a respective patch embedding and a respective token embedding. More specifically, for an image-text pair $(\mathbf{x}_i^v, \mathbf{x}_i^t)$, the similarity $s_{i,lp}$ between a text token embedding $\mathbf{x}_i^t$ and image patch embedding $\mathbf{x}_i^v$ may be determined according to $s_{i,lp} = \mathbf{t}_{il} \cdot \mathbf{v}_{ip}$ where $s_{i,lp} \in \mathbb{R}$ and “ $\cdot$ ” denotes the inner product.

[048] In some implementations, the system 100 may be configured to obtain the attention weights 144 from normalized similarity values, e.g. by normalizing $s_{i,lp}$ to the interval $[0,1]$, e.g. by using a min-max normalization across columns (i.e. patch embeddings): $\hat{s}_{lp} =$

$$\frac{s_{lp} - \min_k s_{lk}}{\max_k s_{lk} - \min_k s_{lk}} \quad (\text{where the image-text pair index “i” has been omitted for clarity}).$$

The system 100 may be configured to generate the similarity matrix 142 (denoted with the symbol “ S ”) by normalizing the relevant similarity values, i.e. $S = (\hat{s}_{jk})_{1 \leq j \leq L, 1 \leq k \leq P}$.

[049] In some implementations, the system 100 may determine the attention weights 144 directly from the similarity matrix S (e.g. each element $\hat{s}_{jk}$ may correspond to one attention weight). In other implementations, the system 100 may be configured to process (sparsify) the similarity matrix S to generate a sparsified similarity matrix, and determine the attention weights 144 from the sparsified similarity matrix. This means that only the most relevant image patches contribute to each language-aware patch embedding.

[050] The similarity matrix may be sparsified in any suitable manner. In particular, the system 100 may be configured to generate the sparsified similarity matrix by processing the similarity matrix S to set a subset of the elements of the similarity matrix S to zero. The similarity matrix may be sparsified by retaining only elements that correspond to similar embeddings according to any suitable criterion, discarding, e.g. setting to zero, elements that correspond to dissimilar embeddings, i.e. those that do not meet the criterion. Thus, the system 100 may be configured to set to zero elements (each element) of the similarity matrix S for which the similarity between the patch embedding and the token embedding corresponding to the element does not meet a similarity criterion. More specially, the similarity matrix may be processed to set to zero elements of the similarity matrix for which the metric of similarity does not satisfy a threshold value.

[051] As an example, the system 100 may be configured to set to zero elements of the similarity matrix S that are below a predefined sparsity threshold value. In this case, the similarity matrix $S = (\hat{s}_{jk})_{1 \leq j \leq L, 1 \leq k \leq P}$ may be sparsified according to $\tilde{s}_{jk} =$

$\begin{cases} \hat{s}_{jk} & \text{if } \hat{s}_{jk} \geq \sigma \\ 0 & \text{otherwise} \end{cases}$, where σ denotes the sparsity threshold value. In this case, the number of zero entries in different rows of the sparsified similarity matrix can be different.

[052] In general, obtaining a sparsified similarity matrix can involve normalizing the elements of the similarity matrix and, for each element of the similarity matrix, comparing the value of the element to the threshold value, setting the value to zero when the value of the element is less than the threshold value.

[053] The system 100 may be configured to compute the alignment weights 144 from the sparsified similarity matrix according to $a_{jk} = \frac{\hat{s}_{jk}}{\sum_{r=1}^R \hat{s}_{jr}}$ where a_{jk} is the weight of patch embedding $\mathbf{v}_k$ for computing the language-aware vision embedding corresponding to token $\mathbf{t}_j$ and R is the number of patch embeddings with non-zero alignment weight. This enables a flexible mapping between a token and arbitrarily many patch embeddings encoding that token in the visual domain, e.g. all image patches corresponding to “dog” can be matched to the token encoding “dog”.

[054] The system 100 may be configured to generate (compute) each language-aware patch embedding by combining the patch embeddings as a weighted average based on the attention weights determined from the sparsified similarity matrix, e.g. the system 100 may be configured to generate (compute) the language-aware embedding $\mathbf{c}_l$ for token $\mathbf{t}_l$ as $\mathbf{c}_l = \sum_{r=1}^R a_{lr} \mathbf{v}_r$.

[055] In some implementations the similarity matrix may be sparsified subject to a criterion that any text token should attend to at least one patch embedding, i.e. to at least one image patch. One possibility of implementing this is by choosing the sparsity threshold σ to be equal to $1/P$ where P is the number of patch embeddings. In this case, the smallest similarity in the (non-sparsified) similarity matrix is $1/P$ (assuming the above-described min-max normalization is used) when all image patches are equally similar since the number of patches is typically constant. This choice of the sparsity threshold (i.e. $\sigma = 1/P$) may conveniently allow the number of patches corresponding to one token to vary considerably between tokens within an image as well as across images. For example, this enables the same class of objects (e.g. “dogs”) to be appropriately represented irrespective of the difference in sizes, scales and shapes across different instances within and across images. This threshold also supports the decoupling of similarities of individual patches to different tokens as it allows different number of zero entries in different rows of the similarity matrix. Thus, whether and how much a patch is similar to a token, has no bearing to how similar it is to a different token

which can be useful e.g. in situations when detailed captions are available (e.g. “large brown dog”) and/or when a single word is represented by multiple tokens. Thus, in some implementations it can be useful to set the threshold value is dependent on $1/P$, where P is the number of patch embeddings.

[056] In some implementations, but not necessarily, the similarity matrix may be sparsified so that a majority of the matrix element values are zero.

[057] As noted above there are various ways of sparsifying the similarity matrix. For example, also or instead of the technique described above, the system 100 may be configured to sparsify the similarity matrix by (pre-)processing the image (in the training data item) to select just a portion of the image. The selected portion of the image can then be used to select the subset of the elements of the similarity matrix to set to zero. This can use, e.g. one or more segmentation masks determined for the image or one or more bounding boxes determined for the image, e.g. of one or more objects in the image. In one possibility, an image may be processed to select part of the image and then the patch embeddings may be generated for patches covering (just) the selected part of the image. Also or instead the selected portion of the image may be used to select the patch embeddings for the similarity matrix. In some implementations, the system 100 may be configured to first sparsify (i.e. pre-sparsify) the similarity matrix based on the selected portion and to then further sparsify the pre-sparsified similarity matrix based on the technique described above.

[058] The training engine 130 is configured to generate (e.g. at each training step) an update 122 comprising parameter updates for updating values of the parameters 140 of the vision encoder neural network 112, and optionally also parameter updates for updating values of the parameters 150 of the text encoder neural network 114 (the text encoder neural network 114 may have been pre-trained.). That is, the training engine 130 trains at least the visual encoder neural network 112 (e.g. at each training step). Generally, the training engine 130 is configured to generate the update 122 based on at least the language-aware patch embeddings 146 and the text token embeddings 134.

[059] More specifically, the training engine 130 is configured to evaluate a local (or “fine-grained”) contrastive objective function 148 over the language-aware patch embeddings 146 and the sequence of token embeddings 134 to generate the update 122 that adjusts the (trainable) parameters 140 of the visual encoder neural network 112. In some implementations both the visual encoder neural network 112 and the text encoder neural network 114 are trained, jointly, by backpropagating gradients of the local contrastive objective function to adjust the parameters of both the visual encoder neural network 112 and

the text encoder neural network 114. The training engine 130 may generate the update 122 by backpropagating gradients of the local contrastive objective function 148 (evaluated over the language-aware patch embeddings 146 and the sequence of token embeddings 134). The training engine 130 may use any appropriate gradient descent optimization algorithm, e.g. Adam or AdamW, or another optimization algorithm, to generate the update 122.

[060] In general, the local contrastive objective function is configured to train the visual encoder neural network 112 and text encoder neural network 114 to encourage a token embedding and the language-aware patch embedding corresponding to the same token (in the associated text) to be similar to one another and dissimilar to the other embeddings in the sequence. That is, the local contrastive objective function encourages “alignment” between a token embedding and the corresponding language-aware patch embedding. The contrastive objective function may be dependent upon a positive example and one or more negative examples. A positive example can comprise a language-aware patch embedding and its corresponding token embedding; a negative example can comprise a language-aware patch embedding and one of the other token embeddings.

[061] In some implementations the (local) contrastive objective function is dependent on a sum, over each token embedding of the sequence of token embeddings, of an objective function having a first term dependent on a similarity (e.g. a value of a similarity metric) between the token embedding and a respective one of the language-aware patch embeddings. The objective function may, e.g. comprise a log function of the first term.

[062] In some implementations the objective function has second term dependent on the similarity (e.g. a value of the same similarity metric). This second term may measure a similarity between the token embedding and (at least) each language-aware patch embedding other than the respective one of the language-aware patch embeddings (in the first term); it may also include a similarity between the token embedding the respective one of the language-aware patch embeddings (in the first term). Also or instead this second term may measure a similarity between the respective one of the language-aware patch embeddings and each other token embedding than in the first term. It may also include a measure of the similarity between the token embedding and the respective one of the language-aware patch embeddings (i.e. the similarity may be between the respective one of the language-aware patch embeddings and each other token embedding).

[063] As an example the first term may be a numerator term in the objective function (and the (local) contrastive objective function may then be a contrastive loss function) and the

second term may be a denominator term in the objective function; or where logs are taken, one term may be subtracted from the other.

[064] In some implementations the contrastive objective function may be symmetrized, e.g. by summing a first version of the objective function comprising the first term and a second term measuring a similarity between the token embedding and each language-aware patch embedding, and a second version of the objective function comprising the first term and a second term measuring a similarity between the respective one of the language-aware patch embeddings and each other token embedding.

[065] As an example a contrastive loss function may be determined as

$$-\log \frac{e^{\text{sim}(c_j, t_j)}}{\sum_{\text{other pairs}} e^{\text{sim}(c_j, t_k)}} \text{ where } \text{sim}(\cdot, \cdot) \text{ denotes a similarity measure such as dot product}$$

similarity, c_i denotes a language-aware patch embedding, t_i the corresponding token embedding, and t_k a different token embedding in the sequence of token embeddings. The denominator sum can run over all the other token embeddings in the sequence, and depending on the implementation can also include the corresponding token embedding (i.e. $e^{\text{sim}(c_j, t_j)}$). The denominator sum may instead be determined as $\sum_{\text{other pairs}} e^{\text{sim}(t_j, c_k)}$; or two terms may be summed, one comprising each version of the denominator sum. There are many different similarity measures that may be used. For example, the measure of similarity may be based on a dot product similarity measure or on a cosine similarity measure.

[066] In implementations the local contrastive objective function is evaluated over the language-aware patch embeddings and the sequence of token embeddings from only a single training data item. That is, the local contrastive objective function may operate over sequences of tokens and patches at the level of each image-text pair and may not require negatives from other image-text pairs (i.e. in some implementations, the local contrastive objective function may not depend on the similarity of a token embedding and a patch embedding from different training data items). This can considerably reduce computation and memory costs over other methods that require samples from the whole batch in order to compute fine-grained losses.

[067] In certain implementations, when the system 100 is configured to process a batch B of the training data items at each training step, the training engine 130 may be configured to evaluate the local contrastive objective function 148 for each training data item and combine the respective evaluations. As an example, the local contrastive objective function applied to a batch B of the training data items may be implemented as a contrastive loss function L_f

given by $L_f = -\frac{1}{2B} \sum_{i=1}^B \left[\frac{1}{L_i} \sum_{j=1}^{L_i} \left(\log \frac{\exp(\text{sim}(c_{ij}, t_{ij})/\tau)}{\sum_{k=1}^{L_i} \exp(\text{sim}(c_{ij}, t_{ik})/\tau)} + \log \frac{\exp(\text{sim}(t_{ij}, c_{ij})/\tau)}{\sum_{k=1}^{L_i} \exp(\text{sim}(t_{ij}, c_{ik})/\tau)} \right) \right]$,

where τ denotes a (learned or predefined) temperature parameter (hyperparameter).

[068] As noted above, in certain implementations when the system 100 is configured to process a batch B of the training data items at each training step, the visual encoder neural network 112 may be configured to determine a global image embedding 136 for each training data item in the batch and the text encoder neural network 114 may be configured to determine a corresponding global text embedding 138 for each training data item in the batch. In these implementations, the training engine 130 may be configured to evaluate a global contrastive objective 152 for the batch of training data items based on the global image embeddings and the global text embeddings for the training data items. More specifically, the global contrastive objective is dependent on a sum, for each training data item, of a global contrastive objective function comprising having a first, e.g. numerator, term dependent on a similarity (similarity metric) between the global image embedding for the training data item and the global text embedding for the training data item. The training engine 130 can (jointly) train (i.e. generate corresponding contributions to the update 122) the visual encoder neural network 112 and text encoder neural network 114 by backpropagating gradients of the global contrastive objective function.

[069] In some implementations the global objective function has a second, e.g. denominator, term dependent on i) the similarity between the global image embedding for the training data item and the global text embedding for each of the other training data items in the batch (and optionally including the global text embedding for that training data item), or ii) the similarity between the global text embedding for the training data item and the global image embedding for each of the other training data items in the batch (and optionally also for that training data item). Optionally the global objective function may be symmetrized in a corresponding way to that previously described.

[070] As an example, the global contrastive objective may be implemented as a global

contrastive loss function L_g given by $L_g = -\frac{1}{2B} \sum_{i=1}^B \left(\log \frac{\exp(\text{sim}(\bar{v}_i, \bar{t}_i)/\tau)}{\sum_{j=1}^B \exp(\text{sim}(\bar{v}_i, \bar{t}_j)/\tau)} + \log \frac{\exp(\text{sim}(\bar{t}_i, \bar{v}_i)/\tau)}{\sum_{j=1}^B \exp(\text{sim}(\bar{t}_i, \bar{v}_j)/\tau)} \right)$ where, as noted above, $\bar{v}_i, \bar{t}_i$ respectively denote the global image embedding and the global text embedding for the i -th training data item in the batch.

[071] In some implementations, the training engine 130 is configured to (jointly) train the visual encoder neural network 112 and the text encoder neural network 114 based on an

overall objective function comprising a weighted combination of the local contrastive objective function and the global contrastive objective function. For example, the overall objective function may be an overall loss function L_{overall} given by $L_{\text{overall}} = \lambda_g L_g + \lambda_f L_f$ where the weights λ_g and λ_f are hyperparameters for controlling the learning process. In general, the overall objective function encourages learning of representations that encode both coarse-grained/global and fine-grained/local information.

[072] In general, the visual encoder neural network 112 and the text encoder neural network 114 may have any appropriate neural network architecture including, e.g., one or more feedforward or convolutional neural network layers, or a transformer neural network subsystem, i.e. a neural network subsystem including one or more transformer blocks or self-attention layers. A transformer block typically includes an attention or self-attention neural network layer followed by a feedforward neural network. An attention, or self-attention, neural network layer is a neural network layer that includes an attention or self-attention mechanism that operates over an attention layer input to generate an attention layer output.

[073] As an example the visual encoder neural network 112 and/or the text encoder neural network 114 may comprise a Transformer neural network. For example, the visual encoder neural network 112 may have an architecture similar to a (standard) vision transformer neural network (e.g. similar to the Vision Transformer (ViT) described in Dosovitskiy et al, arXiv:2010.11929). The text encoder neural network 114 may comprise an encoder-only, encoder-decoder, or decoder-only Transformer neural network (e.g. text encoder neural network 114 may have an architecture similar to the Transformer neural network described in Vaswani et al, Advances in neural information processing systems, 30, 2017). In general, a Transformer neural network can be characterized by having a succession of attention, e.g. self-attention, neural network layers. Each attention neural network layer has an attention layer input for each element of the input and is configured to apply an attention mechanism over the attention layer input to generate an attention layer output for each element of the input. In implementations the attention mechanism computes a similarity between a query and a set of key-value pairs. In implementations one or both (in the case of self-attention) of the query and the set of key-value pairs are determined from the attention layer input.

[074] As an example, in a self-attention neural network layer an input embedding may be used to determine a query vector and a set of key-value vector pairs, that are used to generate an updated embedding comprising a weighted sum of the values, weighted by a similarity function of the query to each respective key. The similarity function may comprise, e.g., a

dot product, cosine similarity, or other similarity measure; the query, keys, and values may all be vectors. For example the attention mechanism may be configured to apply each of a query transformation e.g. defined by a matrix W^Q , a key transformation e.g. defined by a matrix W^K , and a value transformation e.g. defined by a matrix W^V , to the attention layer input for each element of an input sequence X to derive a respective query vector $Q = XW^Q$, key vector $K = XW^K$, and value vector $V = XW^V$ which are used determine an attended sequence for the output.

[075] Once the visual encoder neural network 112 and the text encoder neural network 114 have been trained it may not be necessary to retain both for further use. For example, just the visual encoder neural network may be used in a visual (image) processing task. As another example, both the visual encoder neural network and the text encoder neural network may be used in a visual language model (VLM), which may optionally then be further trained or fine tuned.

[076] Some examples of visual and other tasks that may be performed using the trained visual encoder neural network and/or text encoder neural network are described later.

[077] FIG. 2 is a flow diagram of an example process 200 for training a visual encoder neural network. The process 200 of FIG. 2 may be implemented by one or more computers in one or more locations; for convenience the process is described with reference to FIG. 1.

[078] At an initial step S202, a dataset of training data 120 is obtained. The dataset of training data 120 comprises a plurality of training data items 124. Each training data item comprises an image, which may be still or moving, in 2D or in 3D, and associated text. The associated text defines a sequence of text tokens from a vocabulary of tokens. The tokens can each represent, e.g., words, wordpieces or characters in a natural or computer language.

[079] The process 200 involves performing a plurality of training steps. At least one of the training data items 124 is processed in each training step. In some implementations, in a training step a batch of training data items is processed. Then each training data item can be processed separately and the batch of training data items can also be processed collectively.

[080] At steps 204 to 214, the system 100 processes one of the training data item (steps 204 to 214 may be repeated for each training data item in the batch of training data items). Steps 204 to 214 are described in the following with reference to an example training data item (x_i^v, x_i^t) (using the notation introduced above with reference to FIG. 1). Further, steps 204 to 214 are illustrated in FIG. 3 in which the image x_i^v of the example training data item depicts a dog and a cat, and the associated text x_i^t reads “a picture of a cat and a dog”.

[081] At step 204, pixels of the image $\mathbf{x}_i^v$ in the training data item are processed using the visual encoder neural network 112 (in accordance with the current values of the trainable parameters of the visual encoder neural network 140) to generate a set of patch embeddings 132 for the image (denoted $(\mathbf{x}_{i,1}^v, \mathbf{x}_{i,1}^v, \dots, \mathbf{x}_{i,P}^v)$ where P is the number of patches; it can be seen that in the example of FIG. 3, $P = 6$). Each patch embedding can represent values of the pixels defining the image content of a corresponding patch (region) of the image.

[082] At step 206, the sequence of text tokens $(\mathbf{x}_1^t, \mathbf{x}_2^t, \dots, \mathbf{x}_{L,i}^t)$ of the associated text $\mathbf{x}_i^t$ (where L is the number of tokens in the text $\mathbf{x}_i^t$) of the training data item is processed using the text encoder neural network 114 to generate a sequence of token embeddings representing the sequence of text tokens $(\mathbf{t}_{i,1}, \mathbf{t}_{i,2}, \dots, \mathbf{t}_{i,L_i})$.

[083] Next, the system 100 processes the set of patch embeddings 132 and the sequence of text token embeddings 134 to generate a set of language-aware patch embeddings 146. Each language-aware patch embedding is determined from the set of patch embeddings 134 based on similarities, i.e. values of a metric of similarity, between patch embeddings of the set of patch embeddings and token embeddings of the sequence of token embeddings. As noted above, many possibilities exist for generating the set of language-aware patch embeddings 146. One possibility is described in the following with reference to steps 208 to 212.

[084] In this possibility, at step 208, a similarity matrix 142 representing the similarities between patch embeddings of the set of patch embeddings 132 and token embeddings of the sequence of token embeddings 134 is determined. In some implementations, the process 200 then involves generating the set of language-aware patch embeddings from the set of patch embeddings based on the (non-sparsified) similarity matrix. In other implementations, the similarity matrix is first processed (sparsified).

[085] More specifically, at step 210, the system 100 processes (sparsifies) the similarity matrix to set a subset of the elements of the similarity matrix to zero. As noted above, to this end the system 100 may process the image $\mathbf{x}_i^v$ in the training data item to select a portion of the image and use the selected portion of the image to select the subset of the elements of the similarity matrix to set to zero. Alternatively or in addition, the system 100 can sparsify the similarity matrix by setting to zero elements (each element) of the similarity matrix for which the similarity between the patch embedding and the token embedding corresponding to the element does not meet a similarity criterion. For example, the system 100 may set elements of the similarity matrix to zero for which the metric of similarity does not satisfy a threshold value. More specifically, in some implementations, at step 210, the system 100 may

normalize the elements of the similarity matrix, and then, for each element of the similarity matrix, compare the value of the element to the threshold value and set the value to zero when the value of the element is less than the threshold value (as illustrated in FIG. 3).

[086] In general, generating the set of language-aware patch embeddings from the set of patch embeddings based on the similarity matrix can involve generating each language-aware patch embedding by, for each patch embedding determining a set of attention weights 144, one attention weight for each of the token embeddings, from the elements of the similarity matrix for that patch embedding and for each of the token embeddings. Some of the attention weights may be zero. The process 200 can then determine a language-aware patch embedding, for each token embedding, by combining each of the patch embeddings according to the respective weight for that patch embedding and for the token embedding (i.e. for the corresponding token).

[087] More specifically, at step 212, the system 100 determines the set of attention weights 144. The system 100 may determine, for each patch embedding of the set of patch embeddings, an attention weight for each token embedding. Determining the attention weight for each token embedding can involve, for each patch embedding and for each token embedding of the sequence of token embeddings, determining the attention weight for the patch embedding and for the token embedding from a metric of similarity between the patch embedding and the token embedding.

[088] At step 212, the system 100 can then determine, for each token embedding (token), one the language-aware patch embeddings by combining the patch embeddings according to the attention weights for the token embedding and for the respective patch embeddings (as above, $c_{i,l}$ denotes the language-aware embedding for token $t_{i,l}$). As noted above, the patch embeddings may be combined as a weighted average, weighted by the attention weights.

[089] The process 200 comprises an optional step 214 in which the system 100 determines a global image embedding $\bar{v}_i$ for the image in the training data item by combining the set of patch embeddings for the image, and a global text embedding $\bar{t}_i$ for the associated text in the training data item by combining the text tokens in the sequence of text tokens for the associated text. As noted above, combining the set of patch embeddings for the image can involve average pooling the set of patch embeddings. If necessary the global image embedding and the global text embedding can be mapped to a shared embedding space so that they each have the same dimension, e.g. by applying a linear transformation to one or both of these global embeddings.

[090] At step 216, at least the visual encoder neural network is trained by backpropagating gradients of the above-described local contrastive objective function, evaluated over the language-aware patch embeddings and the sequence of token embeddings, to adjust the (trainable) parameters of the visual encoder neural network. In some implementations both the visual encoder neural network and the text encoder neural network are trained, jointly, by backpropagating gradients of the contrastive objective function to adjust the parameters of both the visual encoder neural network and the text encoder neural network.

[091] In some implementations, at step 216, also the above-described global contrastive objective for the batch of training data items is evaluated (as noted above, the global contrastive objective is dependent on a sum, for each training data item, of a global objective function having a first, e.g. numerator, term dependent on a similarity (similarity metric) between the global image embedding for the training data item and the global text embedding for the training data item). The visual encoder neural network and text encoder neural network can be trained (jointly) by backpropagating gradients of the global contrastive objective function.

In some implementations an overall objective function for training the visual encoder neural network and text encoder neural network comprises a weighted combination of the (local) contrastive objective function and the global contrastive objective function. In these cases, at step 216 both the visual encoder neural network and the text encoder neural network may be jointly trained based on the overall objective function.

[092] Example uses

[093] After training the trained visual encoder neural network can be used to perform an image processing task. Performing the image processing task can comprise processing pixels of a still or moving image to generate the set of patch embeddings for the image, and optionally from them the global image embedding. The set of patch embeddings and/or the global image embedding can be used to perform the image processing task e.g. by using the patch embeddings or by using the global image embedding.

[094] As another example an image processing task may be performed by providing a still or moving image to a visual encoder neural network that has been trained as described above. Pixels of the image are processed using the visual encoder neural network, and the set of patch embeddings and/or the global image embedding for the image is generated and used to perform the image processing task, e.g. as described below.

[095] In general the trained visual encoder neural network can be used to perform an image processing task by itself or with the trained text encoder neural network, as illustrated by some examples below, or it can be used as the neural network backbone for a subsequent task, i.e. it can be used to generate a representation of the image input that is then further processed to generate the desired output.

[096] Similarly the trained text encoder neural network can be used to perform a text processing task either by itself or when used as the neural network backbone for a subsequent task, i.e. it can be used to generate a representation of a text input that is then further processed to generate the desired output.

[097] In some implementations a still or moving (video) image processed by the visual encoder neural network, either during or after training, or both, may be an image that has been captured by a camera, i.e. that has been captured from the real world. Elements of the image data may comprise monochrome or color pixels of the image or video. The image may be a 2D or 3D image. As defined herein an “image” includes a point cloud e.g. from a LIDAR system, and a “pixel” includes a point of the point cloud. Similarly references to a moving image or video include a time sequence of point clouds. Objects in the image or video may comprise objects, e.g. physical objects, represented by the image or video.

[098] As one example, the trained visual encoder neural network and text encoder neural network can be used to perform a still or moving image classification task (zero-shot). For example to classify an image into one of a plurality of classes, e.g., as a pickup truck, car, or van, the global text embedding can be determined for each of a set of words or sentences that describe the image as belonging to a different respective class, e.g. “this is a photograph of a pickup truck”, and so forth. The global image embedding can be determined for the image, and the class of the image can be determined from the word or sentence that has a global text embedding that is most similar to the global image embedding. A similar approach may be used to classify actions in moving images, e.g. gestures; and to perform a multi-label classification.

[099] This approach may also be used to retrieve an image from a database, by comparing the global text embedding of an index word or sentence with the global image embedding of each image in the database to find the closest match. Correspondingly a similar approach may be used to perform an image captioning task, by comparing the global image embedding of an image with the global text embedding of each word or sentence image in the database to find the closest match. As another example, the patch embeddings and/or global image embedding of a target image and other images may be compared, e.g. to determine a value

representing a similarity between the target and another images, e.g. as part of an image search task, to identify and retrieve one or more images that are similar to a target image.

[0100] As another example the trained visual encoder neural network may be used to perform (unsupervised) semantic segmentation of an image, e.g. by applying a clustering algorithm such as k-means clustering to a (normalized) output of the visual encoder neural network, e.g. to the patch embeddings, optionally processed by the previously mentioned non-linear layer. This can cluster or classify each patch into one of a set of categories, e.g. by determining defining a score for each category of a set of possible categories. The semantic segmentation can also be performed based on a text label, e.g. to segment the image according to a description, e.g. object definition, in the text label. This can be done by determining a similarity of each patch embedding, e.g. a cosine similarity, to the global text embedding of the text label to identify those that match with greater than a threshold level of similarity (instead of clustering). In general the output of any dense prediction task, e.g. a semantic or other segmentation task, may comprise a value for each pixel or patch that defines the desired output, e.g. a value for each pixel or patch that defines a class or category to which the pixel or patch belongs for a segmentation task, or a depth value for a depth estimation task, and so forth.

[0101] Many other image processing tasks can be performed using the set of patch embeddings and/or the global image embedding for an image, typically by adding a neural network head to a neural network backbone comprising the visual encoder neural network. In general techniques for processing image representations, e.g. patch representations, to perform image processing tasks are well known. For example many examples have previously been described for Vision Transformer neural networks (that also generate patch representations), and these may also be used with the trained visual encoder neural network. This is another way in which an image classification task or semantic segmentation task as previously described may be performed.

[0102] Such an approach may be used to perform an image processing task comprising one or more of: image segmentation, e.g. semantic segmentation or instance segmentation; depth prediction; keypoint prediction; pose estimation, e.g. 3D pose estimation; surface normal estimation, e.g. by determining a vector in 2D or 3D; or object detection, including object tracking. In general any prediction task may be performed, e.g. by determining a scalar or vector value for each patch of an image, e.g. using a head neural network. Other types of task may be performed in the same way, e.g. a curvature or other shape estimation task, a task that involves identifying aspects of an image using color, a counting task that involves counting

objects or objects of a particular type, a task that involves understanding spatial relationships between objects or object attributes, and so forth.

[0103] Some other tasks that can be performed using the trained visual encoder neural network include image enhancement, image colorization, and image super-resolution, i.e. to generate output data that comprises an enhanced, colorized, or super-resolution version of an input image.

[0104] Some other tasks that can be performed using the trained visual encoder neural network and text encoder neural network include multimodal tasks that involve processing a combination of an image and text to generate an output that performs the image processing task. The output can be generated from a neural network that processes both the patch embeddings and the token embeddings.

[0105] Thus some implementations of the method involve receiving a sequence of token embeddings from a trained text encoder neural network, e.g. pre-trained or trained as described above, and processing the one or more patch embeddings and/or the global image embedding and the sequence of token embeddings to perform an image processing task defined by a sequence of text tokens encoded by the sequence of token embeddings.

[0106] One example is a visual ground task that takes as input an image and text in a natural language and that generates an output that identifies or locates the most relevant object or region in an image, in particular by processing the patch embeddings and the token embeddings.

[0107] Another example involves generating an output that requires reasoning, e.g. spatio-temporal reasoning, to respond to a natural language query input, e.g. relating to a moving image (video). For example such a query may require predictive reasoning (“what will happen next”), counterfactual reasoning (“what would happen in a different circumstance”), explanatory reasoning (“why did something happen”), or causal reasoning generally. For example the trained neural networks can be used to detect objects in the video frames and provide information relating to the detected objects in response to a query. The query may comprise, for example, a request for a prediction of a future event or state relating to one or more of the objects (e.g. “will objects X and Y collide?”), or a request for conditional or counterfactual information relating to one or more of the objects (e.g. “what event would [not] happen if object X is modified, moved or absent?”), or a request for analysis of the video frames to determine a property or characteristic of one or more of the objects (e.g. “how many objects of type Z are moving?”). The output may, for example, be in the form of a yes/no answer, or may define a probability distribution over a set of possible answers; or

the response may define the location of an object. Such systems can be used to predict whether or not two objects will collide, or how this may be avoided. The output may be useful by itself or it may be used to provide a warning and/or to control motion of one or more of the objects.

[0108] As one example, in an image segmentation task the patch embeddings may be processed to assign a categorical value defining a category for the patch, or a value representing a probability that the patch belongs to a particular category. The category may represent an object or type of object or (for video) an action or type of action. For example in a semantic segmentation task the patch values may identify a type or category of object and in an instance segmentation task the values may (also) distinguish between different instances of the same category of object. More generally a value can distinguish between an object (or action) and image background, and the set of patch embeddings for an image can, e.g., perform an object localization, detection, or tracking task, e.g. for gesture recognition, i.e. recognition of gestures that are performed by entities depicted in a video.

[0109] Also or instead the patch embeddings may be processed to determine data representing one or more bounding boxes or other location data for an object or type of object in the processed image or, for moving images, location data for an action or type of action in the processed image. Such location data may comprise, e.g. data defining coordinates of a bounding box or region for one or more objects represented in the image. Such a bounding box or region may be defined in two, three or more dimensions (time counting as a dimension). Such location data may contribute to higher level tasks, e.g. to object tracking across video frames.

[0110] Merely as some further illustrative examples, object segmentation may be used to segment medical images, to label patches of an input medical image in accordance with whether they show a region of a human or animal body in which a particular medical condition is present. An object segmentation may be used to provide an input to a control system of a mechanical agent, such as a robot or vehicle operating in a real-world environment. The detected objects may be, e.g., obstacles or paths upon which the mechanical agent can move, and may be used by the control system e.g. to make decisions on how to accomplish a task performed by the robot, or for controlling the direction or speed of movement of the agent.

[0111] More generally the set of patch embeddings and/or global image embedding can be used to perform an agent control task, where the patch embeddings represent an observation of an environment, e.g. a real world environment, are processed to generate an output that

defines an action to be performed by the agent, in particular to perform a task. The agent can be a mechanical agent, e.g. a robot or vehicle, controlled to perform actions in the real world environment, in response to the observations, to perform the task, e.g. to manipulate an object or to navigate in the environment. Thus the agent can be, e.g., a real-world or simulated robot; as some other examples the agent can be a control system to control one or more machines in an industrial facility.

[0112] As a more particular example the set of patch embeddings and/or global image embedding may be used to provide an input to a control system of a mechanical agent, such as a robot or vehicle operating in a real-world environment. The control system may provide an output that controls the operation of the robot or vehicle to perform a task such as manipulating an object in the environment or moving in the environment. The set of patch embeddings and/or global image embedding may be used to, e.g. to detect objects for the robot to manipulate, or obstacles or paths upon which the mechanical agent can move, and may be used by the control system e.g. to make decisions on how to accomplish a task performed by the robot, or for controlling the direction or speed of movement of the agent.

[0113] As another example, in a (monocular) depth prediction task the patch embeddings may be processed to obtain a scalar patch value representing an estimated depth value for the patch, e.g. a distance of the patch in a depth or z-direction from an x-y image plane or camera viewpoint. Or the patch embeddings may each define a depth distribution, e.g. a probability distribution over discrete depth value buckets. The depth values can define a depth map for the image.

[0114] As another example, in a keypoint prediction task the patch embeddings may identify keypoints in the image, e.g. by labelling a patch as a keypoint or as one of multiple keypoints. A set of values for the patches can thus label keypoints in the image that may, e.g. define landmarks of an object represented in the image, e.g. the positions of body joints for a human.

[0115] As another example, in a pose estimation task the patch values may map the patches to a 3D surface, e.g. of a human body or face. Or the patch values may estimate a 6D pose representing translation and orientation components of an object in the image, e.g. in quaternion form. The set of patch values for the image can estimate the pose of one or more objects in the image.

[0116] As another example, in a surface normal estimation task the patch values for the image may comprise a vector in, e.g., three dimensions defining a surface normal. The set of patch values for the image can provide a surface normal map for one or more objects in the image, e.g. for use in an augmented reality or other application.

[0117] In some other applications one or both of trained visual encoder neural network and the trained text encoder neural network may be used as an image encoder or as a text encoder for a multimodal model such as a Visual Language model (VLM). In such applications parameters of the trained visual encoder neural network and/or the trained text encoder neural network may be frozen, or they may be further trained, e.g. fine tuned, either on a general corpus of training data or for one or more specific tasks using appropriate datasets (of which many are available).

[0118] Further, while the generation of the language-aware patch embeddings has been described above (with reference to Figures 1 to 3) in the context of training the vision encoder (and optionally the text encoder), it is to be understood that the language-aware patch embeddings may also be useful for applications other than the described training (e.g. for VLM applications described below). Thus, in some implementations, the trained visual encoder neural network and the trained text encoder neural network may be used to process an input image and an associated text to generate a corresponding set of the language-aware patch embeddings (i.e. in these cases the language-aware patch embeddings may be generated in a similar manner as described above with reference to Figures 1 to 3).

[0119] Multimodal, e.g. visual language models

[0120] In general a multimodal machine learning model has a multimodal input configured to receive a first multimodal input and a second multimodal input. As used herein a “modality” refers to a type of data, and thus a multimodal machine learning model is one that can process multiple different types of data.

[0121] The first multimodal input may comprises a text input to receive a sequence of text. The second multimodal input may be configured to receive a different type of input data, e.g. it may comprises a visual input to receive an image or video. Alternatively the second multimodal input may be configured to receive, e.g., audio data representing values of an audio waveform, e.g. instantaneous amplitude data or time-frequency domain data; or data representing observations (not necessarily visual) of an environment with which an agent controlled by the multimodal machine learning model interacts. In some implementations there may be more than two different multimodal inputs, each configured to receive a different type of data.

[0122] The multimodal machine learning model may be configured to jointly process an encoded version of the text and an encoded version of the second multimodal input, e.g. of the image or video, to generate a model output that defines a result of a machine learning

task. A few examples of machine learning tasks that can be performed by such a multimodal machine learning model are described later.

[0123] The text received may comprise text in one or more natural languages, or text in a computer language, or both. The computer language may be any formal language used to communicate with a computer, e.g. a markup language, or a command or configuration language, or a data exchange language such as JSON, or a programming language. The text may be received, e.g., as a series of encoded characters, e.g. UTF-8 encoded characters; such “characters” can include Chinese and other similar characters, as well as logograms, syllabograms and the like.

[0124] The multimodal machine learning model can include a text encoder that processes the sequence of text to represent the text as a series of text tokens from a vocabulary of text tokens, e.g. that each represent words, wordpieces or characters in a natural or computer language.

[0125] Where the second multimodal input comprises an image or video it may comprise image data defining color or intensity values for pixels of a still or moving image in one, two, or three dimensions. As used herein “image” includes a LIDAR point cloud, and the image data may also or instead define the locations of points of a still or moving point cloud. As another example, an image or video received by the second multimodal input may comprise a neural 3D representation, e.g. that represents a 3D scene as a set of latent feature vectors, e.g. a neural radiance field representation. The multimodal machine learning model can include second multimodal input encoder that processes the second multimodal input, e.g. using one or more convolutional, attention, fully connected, or recurrent layers, to generate the encoded version of the second multimodal input. In general such an encoder may implement any form of encoding appropriate for the type of data to be encoded. Merely as an example, where the second multimodal input comprises an image or video this may be encoded, e.g., as features for each of a set of patches that tile the image, or as a sequence of visual tokens selected from a vocabulary of visual tokens, or as a representation of distinct objects in the visual input. Such visual tokens may, but need not be, interleaved with text tokens processed by the model.

[0126] The model output may comprise any form of output appropriate to the machine learning task performed by the multimodal machine learning model. For example the model output may comprises text in a natural or computer language that defines a result of the task, e.g. for tasks such as image captioning, visual question answering, or object detection or instance segmentation. Also or instead the model output may comprise data defining an image, video or audio object, e.g. in a generative task; or the model output may comprise

non-textual action selection data for selecting an action to be performed by an agent controlled by the model. As another example the model output may also or instead define an intermediate step to be performed during the task, e.g. a call to a software API for a software tool that is used when performing the task; the multimodal input may then receive an output from the software tool that is used to generate a final model output that performs the task. A few particular examples of model output are given later.

[0127] Such a multimodal model can be trained using very large (but possibly noisy) datasets in which text is paired with an image and/or with one or more other types of data, e.g. audio data, or data relating to the operation of an agent acting in an environment to perform a variety of tasks. Such a model is can be trained, e.g., using self-supervised learning. The pairing can often be imperfect, and the training dataset can, but may not, include any actual examples of a particular task to be performed, but nonetheless an ability to perform a particular task can emerge. There are many examples of suitable, publically available training datasets.

[0128] Some example multimodal machine learning models with which the techniques described herein may be used include: Flamingo (Alayrac et al. arXiv:2204.14198); ALIGN (Jia et al., arXiv:2102.05918); PaLI (Chen et al. arXiv:2209.06794); and PaLI-X (Chen et al. arXiv:2305.18565).

[0129] In some implementations, the second multimodal input can include an observation characterizing an environment of an agent performing a task, e.g. a mechanical agent or software agent. The observation may characterize the environment at a particular time step and the model output may define one or more actions to be performed by the agent at the time step. For example each action may be expressed as a sequence of text, e.g. as one or more characters such as letters and numbers, that represents the action, or as text that defines a low-level “skill” from a set of skills; or the model output may, e.g., define parameters of a probability distribution from which an action is selected. Optionally the text received by a text input may include text describing the task to be performed. Optionally the text input may include a description of one or more actions performed at a preceding time step. Where the agent is a software agent the model output may comprise a text output for calling a software API at a time step, and the model input at a subsequent time step, e.g. the text input, may comprise a response from the software agent, e.g. from the API.

[0130] Some examples of multimodal machine learning models controlling an agent, and with which the techniques described herein may be used, are described in: PaLM-E (Driess et

al. arXiv:2303.03378); RT-1 (Brohan et al. arXiv:2212.06817); and RT-2 (Brohan et al. arXiv:2307.15818).

[0131] Such a multimodal machine learning model can have an audio input, or an agent action input to receive agent action data representing an action of an agent performing a task in an environment. Data received in this way may be jointly processed with data from a text input and from a second multimodal input to generate the model output.

[0132] Such a multimodal machine learning model has a multimodal input and can, in implementations, perform a range of different tasks. However in implementations not every task that the model performs requires a multimodal input, e.g. a task to generate an image from a text description of the image, or an image captioning task. In some implementations, after training, the text input can be used to specify a particular task that is to be performed by the multimodal machine learning model, e.g. by providing a “prompt” to the model describing the task to be performed or giving an example of the task as a prompt to the model. Such prompts may optionally be included in the training data.

[0133] The multimodal machine learning model is configured to process the multimodal input in accordance with the trainable parameters of the multimodal machine learning model, to generate a model output that defines a result of one or more machine learning tasks. A training system can include a training engine to train the multimodal machine learning model, i.e. to update values of the trainable parameters, to perform the machine learning task(s), using training data items stored in one or more training datasets.

[0134] In general each training data item comprises multimodal data for use in training the multimodal machine learning model, e.g. using a self-supervised training objective. As another example such a multimodal machine learning model can also or instead be trained using a reinforcement learning objective, e.g. when the model is used to control an agent to perform a task.

[0135] There are many different types of self-supervised objective function that may be used. As one example the model may be trained using a softmax cross entropy loss, e.g. using language model style teacher forcing with a softmax cross entropy loss. As another example the model may be trained with an autoregressive negative log likelihood (NLL) loss, such as $-\sum_{l=1}^L \log p(y_l | y_{<l}, x_{\leq l})$ for a multimodal input comprising a sequence of text encoded as L tokens with the l th text token y_l conditioned on preceding second modality inputs $x_{\leq l}$, such as one or more images or videos, and conditioned on preceding text tokens $y_{<l}$. As another example the model may be trained with a masking loss, e.g. a loss that requires the model to

predict masked-out data such as masked out text tokens. As another example the multimodal machine learning model can be trained using a self-supervised objective function that comprises a contrastive loss function (one that is dependent upon a positive example and one or more negative examples).

[0136] There are, similarly, many different types of reinforcement learning objective function that may be used.

[0137] Each training data item may comprise, e.g., an example sequence of text and an example of the second modality input, e.g. an example image or video; in general these are semantically related to one another (but not always, as the training dataset may be noisy). As an example, matched, text and image or video data and may be obtained from web pages, e.g. from images or videos and their corresponding alt-text (text from the HTML or XHTML alt attribute); or from web pages where images or video and text are interleaved with one another. One example of such a dataset is WebLI (Web Language Image, Chen et al. arXiv:2305.18565v1). Training datasets for other types of second modality input can similarly be obtained from web pages. Such training datasets can be large, e.g. $> 10^7$, 10^8 or 10^9 items.

[0138] Also or instead, smaller but more specialized training datasets can be used, e.g. to fine tune a model for a particular task or tasks. A few examples for visual tasks are the Visual Genome dataset for Visual Question Answering (Krishna et al., arXiv:1602.07332); Objects365 (Shao et al., “Objects365: A large-scale, high-quality dataset for object detection”, IEEE/CVF international conference on computer vision, pages 8430–8439); Open Images V4 (Kuznetsova et al., arXiv:1811.00982); the SBU dataset (Ordonez et al. “Im2Text: Describing Images Using 1 Million Captioned Photographs”, NeurIPS 2011); the Conceptual Captions datasets, e.g. V1 (2M images) or V2 (10M images) (Sharma et al., “Conceptual Captions: A Cleaned, Hypernymed, Image Alt-text Dataset For Automatic Image Captioning”, ACL 2018); and Kinetics for video (Kay et al., arXiv:1705.06950). An example task-specific training dataset for audio data is AudioSet (Gemmeke et al., “Audio set: An ontology and human-labeled dataset for audio events,” ICASSP, IEEE, 2017, pp. 776–780). An example task-specific training dataset for agent (robot) control is described in Ebert et al., arXiv:2109.13396.

[0139] Example tasks for multimodal models such as VLMs

[0140] In general a multimodal machine learning model can be trained to perform any sort of machine learning task or tasks. After the multimodal machine learning model has been trained it can be deployed for use in performing the machine learning task(s). For instance,

the machine learning model can be deployed in an environment that enables users to provide requests for the machine learning model to process specified multimodal inputs to generate corresponding model outputs. Users can provide the requests, e.g., by way of a user interface or through an application programming interface (API). The requests can be transmitted from a user device (e.g., over a data communication network, e.g., the internet) to one or more computers implementing the machine learning model, e.g., in a data center. The machine learning model can process multimodal inputs specified by user requests to generate corresponding model outputs, and then transmit the model outputs to user devices (e.g., over a data communication network).

[0141] In some implementations, after training, a particular task that is to be performed by the multimodal machine learning model can be described by part or all of the sequence of text in the multimodal input to the model. For example in a multimodal input that includes an image, video, or audio item such a prompt might specify “Generate a caption”, “Generate a description”, “Answer the following question: [about the image, video, or audio item]”, or “Detect a person”. Where the model is used for an agent control task a prompt may define “Take the knife out of the drawer”, or “Q: What action should the robot take to take the knife out of the drawer?”. Also or instead such a prompt may give one or more examples of a task to be performed. A multimodal machine learning model can be trained on multiple natural and/or computer languages and the prompt may then specify a language to use.

[0142] A few examples of some machine learning tasks that can be performed by a model trained as described herein follow.

[0143] For some tasks the second modality input represents an image or video as previously described, e.g. from a camera or other imaging device that captures the image or video from a real-world environment, and/or audio, e.g. audio data such as speech or other sounds captured from a real-world environment. In general the tasks described below may be tasks that require spatial awareness or other context from the image, video, or audio item. For example, a prompt may ask “What is the object in the top left corner?”, or “What was the answer to the spoken question?”.

[0144] As one example the task may comprise an object or action detection task. A task-specific training data item may comprise an image, video, or audio item containing one or more objects or actions, and a sequence of text. The sequence of text may describe or otherwise label the object(s) or action(s) and (for an image or video) may include text giving bounding box coordinates for the object(s) or action(s). After training, when the model is used in inference, the model output 122 may comprise or represent text that describes or

otherwise labels detected object(s) or action(s) in the second modality input, and may (for an image or video) include bounding-box coordinates for the detected object(s) or action(s), e.g. "10 20 90 100 cat 20 30 100 100 dog".

[0145] As another example the task may comprise a classification task, e.g. an object or action classification task. A task-specific training data item may comprise an image, video, or audio item containing one or more objects or actions and a sequence of text. The sequence of text may describe or otherwise classify the object(s) or action(s). After training, when the model is used in inference, the model output may comprise data, e.g. text, that classifies the object(s) or action(s) in the second modality input into one of a plurality of classes.

[0146] As another example the task may comprise an image, video, or audio item describing task, e.g. a captioning task (which, as used here, includes an audio description task to explain what is happening in a video). A task-specific training data item may comprise an image, video, or audio item and a sequence of text describing the image, video, or audio item. After training, when the model is used in inference, the model output may comprise data, e.g. text, describing an image, video, or audio item in the second modality input. For example the model output may provide a caption or description for a second modality input item, or it may count objects in the second modality input item, or it may provide some other form of description of the second modality input item.

[0147] As another example the task may comprise an image, video, or audio question-answering task. A task-specific training data item may comprise an image, video, or audio item and a sequence of text that describes the image, video, or audio item. After training, when the model is used in inference, the model output may comprise data, e.g. text, that answers a question about the second modality input specified in a prompt sequence of text, e.g. as described above. This may be used, e.g., to answer questions about visual plots and charts or about sounds.

[0148] As another example the task may comprise a character or word recognition task, e.g. an OCR (optical character recognition) task. A task-specific training data item may comprise an image, video, or audio item and a sequence of text that includes text that is depicted in the image or video, or that is represented as speech in the audio item. After training, when the model is used in inference, the model output may comprise text that represents characters or words in the second modality input, e.g. in a natural language.

[0149] As another example the task may comprise a still or moving image or audio generation task. A task-specific training data item may comprise an image, video, or audio item and a sequence of text that describes the image, video, or audio item. After training,

when the model is used in inference, the model output may comprise data for an image, video, or audio item, e.g. image data defining values for pixels of a still or moving image or audio data representing values of an audio waveform, and the sequence of text in the multimodal input to the model may describe or characterize the image, video, or audio item to be generated.

[0150] As another example the task may comprise a computer language text generation task. A task-specific training data item may comprise an image, video, or audio item and a sequence of text in a computer language for generating the image, video, or audio item. After training, when the model is used in inference, the model output may comprise text in the or another computer language for generating or rendering an image, video, or audio item in the second modality input, e.g. a web page, plot, or chart.

[0151] In another example of a computer language text generation task a task-specific training data item may comprise an image, video, or audio item and a sequence of text in a computer language for performing a task in relation to the image, video, or audio item, e.g. a data processing task that involves analyzing the content of the image, video, or audio item to provide a result of the analysis or, e.g., a search to search for information relating to the content of the image, video, or audio item. The computer language in the model output may comprise computer language for invoking a function or calling one or more external APIs. Merely as one example, such an output may be formatted as a JSON object. As previously, the sequence of text in the multimodal input may define the task to be performed and the second modality input may comprise, e.g. an image, video, or audio item in relation to which the task is to be performed, e.g. a task that involves manipulation of particular types of data that may benefit from access to an API such as mathematical data, date/time related data, scientific data, recent data that may post-date training of the model (that may be accessed by a search function or API), and so forth. After training, when the model is used in inference, the model output may comprise text in the or another computer language for performing a task, e.g. as described above, in relation to an image, video, or audio item in the second modality input. The method may then include using the text in the computer language to perform the task.

[0152] In general where the model output comprises text this may be provided as speech representing the text.

[0153] In some implementations the machine learning task comprises an agent control task in which the agent interacts with an environment to perform the agent control task. In these implementations the multimodal input includes an observation characterizing the

environment. For example the multimodal input can include a sequence of text that defines the task to be performed by the agent and the second modality input can represent an image, video, audio, or other observation of the environment, e.g. captured by a camera or other imaging device, or by a microphone, from a real-world environment. A task-specific training data item may comprise a sequence of text representing one or more actions of the agent, and a second modality input representing an observation of the environment. After training, when the model is used in inference, the model output comprises an action selection output, e.g. including text, that is used to select one or more actions to be performed by the agent in the environment in response to the observation. As an illustration the model output 122 may define an action as text such as “A: 132 114 128 5 25 156”, that can be converted into a control signal for a mechanical agent, such as a robot, e.g. “ $\Delta T = [0.1, -0.2, 0]$ $\Delta R = [10^\circ, 25^\circ, -7^\circ]$ ”. As another example the action selection output may also or instead define one or more low-level skills, e.g. from a vocabulary of previously learnt skills. As before, the sequence of text in the multimodal input to the model may describe the task to be performed, e.g. “What action should the robot take to [perform task]”.

[0154] In some agent control implementations, the environment is a real-world environment and the agent is a mechanical agent interacting with the real-world environment, e.g., a robot or an autonomous or semi-autonomous land, air, or sea vehicle operating in or navigating through the environment, and the actions are actions taken by the mechanical agent in the real-world environment to perform the task. For example, the agent may be a robot or other mechanical agent interacting with the environment to accomplish a specific task, e.g., to locate or manipulate an object of interest in the environment or to move an object of interest to a specified location in the environment or to navigate to a specified destination in the environment. In these implementations, the observations may include, e.g., one or more of: images, object position data, and sensor data to capture observations as the agent interacts with the environment. The actions may define control signals to control the robot or other mechanical agent, e.g., positions, torques, or other control signals for the parts of the mechanical agent, or higher-level control commands.

[0155] In some agent control implementations the agent can be a software agent, i.e. a computer program, configured to perform a task. Some examples where the agent is a software agent now follow.

[0156] As one example the environment may be an integrated circuit design and the task may be a routing task for routing interconnection lines of the integrated circuit. The observations may be of component positions and/or interconnections, and the actions may comprise

component placing or interconnect routing actions. An integrated circuit with interconnection lines routed as determined may then be fabricated.

[0157] As another example the environment may be a real-world computing environment and the task may be to manage the distribution of jobs or tasks across computing resources e.g. on a mobile device and/or in a data center. The observations may include observations of computing resources such as compute or memory capacity, or Internet-accessible resources, or that relate to the operation of the computing resources in processing the jobs or tasks; and the actions may include assigning jobs or tasks to particular computing resources.

[0158] As another example the environment may be a real-world computing environment and the task is to manage the processing, e.g. by one or more real-world servers, of a queue of continuously arriving jobs. The observations may comprise observations of the times of departures of successive jobs, or the time intervals between the departures of successive jobs, or the time a server takes to process each job, or the arrival times, or time intervals between the arrivals, of successive jobs, or data characterizing the type of job(s). The actions may comprise actions that allocate particular jobs to particular computing resources.

[0159] As another example the environment may comprise a real-world computer system or network and the task may be to maintain security of the computer system or network. The observations may comprise any observations characterizing operation of the computer system or network, and the actions may comprise actions to control the operation e.g. to limit or correct abnormal or undesired operation e.g. because of the presence of a virus or other security breach.

[0160] As another example the environment may comprise a data packet communications network environment, and the task may be to route packets of data over the communications network. The actions may comprise data packet routing actions and the observations may comprise, e.g., observations of a routing table which includes routing metrics such as a metric of routing path length, bandwidth, load, hop count, path cost, delay, maximum transmission unit (MTU), and reliability.

[0161] In some agent control implementations the agent may be a human agent and the environment may be a real-world environment. For example the agent can be a human user of a digital assistant such as a smart speaker, smart display, or some other device that is used to instruct the user to perform actions. The task may be any real-world task that the user wishes to perform. The observations may be obtained from an observation capture subsystem, e.g. a monitoring system such as a video camera or sound capture system, to capture visual and/or audio observations of the user performing the task. The actions may

comprise instructions in the form of, e.g., text, image, video, or audio data such as speech, that guide the user in performing the task.

[0162] The performances of three example neural networks trained using the system 100 of FIG. 1 have been experimentally investigated for multiple image processing tasks. In the example neural networks, the vision encoder neural network is a Vision Transformer (ViT) (as described in Dosovitskiy et al., 2020, arXiv:2010.11929), and the text encoder neural network is a Transformer (as described in Vaswani et al., 2017, Advances in neural information processing systems). More specifically, the vision encoder of the first example neural network is a ViT-Base image encoder model with 12 layers, 768 width, 12 attention heads and a patch size of 32x32 (“ViT-B/32” for short). The vision encoder of the second example neural network is also a ViT-Base image encoder model with 12 layers, 768 width, and 12 attention heads but with a patch size of 16x16 (“ViT-B/16” for short). The vision encoder of the third example neural network is a ViT-Large image encoder model with 24 layers, 1024 width, 16 attention heads and a patch size of 14x14 (“ViT-L/14” for short). The text encoder neural network of example neural networks is a Transformer model having an architecture with 12 layers, 768 width and 12 attention heads. Linear adapters $g_v(\cdot)$ and $g_t(\cdot)$ are used to project the patch and the token embeddings respectively to a shared embedding space of dimensionality 512.

[0163] The example neural networks are (pre-)trained using the ALIGN (Jia et al., “Scaling up visual and vision-language representation learning with noisy text supervision”, International conference on machine learning, 2021), JFT (Sun et al., “Revisiting unreasonable effectiveness of data in deep learning era”, Proceedings of the IEEE international conference on computer vision, 2017; Zhai et al., arXiv:2111.08276, 2022) and LTIP (Long Text & Image Pairs) (Alayrac et al., “Flamingo: a visual language model for few-shot learning”, Advances in Neural Information Processing Systems, 2022) datasets according to the above described process 200. ALIGN has 1.8 billion images paired with noisy alt-text, JFT has of 4 billion images semi-automatically annotated with a class-hierarchy of 30k labels, and LTIP has 312 million higher-quality image-text pairs with richer image captions. For JFT, the hierarchical label structure is flattened and all assigned labels are used to describe the image. A multi-step training strategy is used in which sampling batches from each of the 3 large datasets is alternated; the gradient updates are then performed by aggregating the gradients from computing the loss on one batch from each of the datasets.

[0164] For each example neural network, comparative models have been trained based on CLIP (Radford et al., arXiv:2103.00020, 2021), FILIP (Yao et al., arXiv:2111.07783, 2021), PACL (Mukhoti et al., “Open vocabulary semantic segmentation with patch aligned contrastive learning”, Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023), MGCA (Wang et al., “Learning robust global representations by penalizing local predictive power”, Advances in Neural Information Processing Systems, 2022) and GLoRIA (Huang et al., “A multimodal global-local representation learning framework for label-efficient medical image recognition”, Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021) using the same training datasets, architecture and number of training steps when training with different objectives.

[0166] More specifically, the training images have been resized to the 224x224 resolution and the training text has been tokenized with a 32k vocabulary sentencepiece tokenizer (Kudo and Richardson, arXiv:1808.06226, 2018) while keeping a maximum number of 55 tokens for each caption. All models (i.e. the example neural networks and the comparative models) have been using the AdamW optimizer, a cosine learning rate schedule with linear warm-up of 2500 steps and weight decay regularization. A sweep over the learning rate and weight decay values in the following ranges has been performed: learning rate in $[7e-4, 9e-4, 1.1e-4]$ and weight decay in $[0.1, 0.2, 0.3]$. A batch size of 16348 (except for GLoRIA for which we use 4096 batch size) has been used. The ViT-B models are (pre-)trained for 200k steps (≈ 3.2 billion data points) and the ViT-L models for 250k steps (≈ 4.1 billion data points).

[0167] The example neural networks have been trained using the above described overall objective (i.e. a weighted combination of the local contrastive objective function and the global contrastive objective function) where the global loss weight has been set to $L_g = 0.5$ and the local loss weight L_f has been swept over $[0.5, 1.0, 5.0, 10.0]$. Further, a learned temperature parameter τ has been used.

[0168] The hyperparameters of the comparative models have been selected based on the referenced publications (except for PACL where a learnable temperature parameter has been included in the loss as it has been found that this significantly improves the performance).

[0169] Experimentally obtained performance values for the example neural networks and the comparative models are described with reference to FIGS. 4 to 10. In these Figures, the example neural networks trained according to the process 200 are labelled “SPARC” (short for “SPARse fine-grained Contrastive alignment”).

[0170] FIG.4 shows tables 400 and 402 containing performance values (top-1 accuracy in %) of the example neural networks and the comparative models on a zero-shot image classification task. More specifically, the models have been tested on zero-shot classification on ImageNet (Russakovsky et al., “Imagenet large scale visual recognition challenge”, International journal of computer vision, 2015) and a number of datasets testing for specific capabilities like robustness to perturbations and various distribution shifts; in particular, ImageNetV2 (Recht et al., “Do imagenet classifiers generalize to imagenet?”, International conference on machine learning, PMLR, 2019), ImageNet-R (Hendrycks et al., “The many faces of robustness: A critical analysis of out-of-distribution generalization”, Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021), ImageNet-C (Hendrycks and Dietterich, arXiv:1903.12261, 2019), ImageNet-A (Hendrycks et al., cs.LG/1907.07174, 2019) and ImageNet-Sketch (Wang et al., “Learning robust global representations by penalizing local predictive power”, Advances in Neural Information Processing Systems, 2019). The performance of the model has been evaluated following a similar protocol to (Radford et al., “Learning transferable visual models from natural language supervision”, International conference on machine learning, PMLR, 2021). Table 400 contains results for one prompt per example (i.e. the class label), and table 402 contains results for when prompt ensembling is used.

[0171] It can be seen from FIG. 4 that the trained example neural networks outperform (or match) the comparative methods in all settings and across different ViT architectures. More specifically, the trained example neural networks show very effective information encoding from larger patches as exhibited by the significant improvements over the comparative models for ViT B/32, especially on ImageNet-R, -C, -A and -Sketch demonstrating robustness to perturbations and adversarial examples. Notably, the improved performance of the example neural network is preserved when prompt ensembling is used.

[0172] FIG. 5 shows a table 500 containing performance values of the example neural networks and the comparative models on cross-modal retrieval tasks, in particular on a zero-shot image-to-text task and a text-to-image retrieval task on Flickr30k (Plummer et al., “Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models”, Proceedings of the IEEE international conference on computer vision, 2015) and MSCOCO (Lin et al, “Microsoft coco: Common objects in context”, Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014) datasets. From table 500, it can be seen that the example neural networks outperform (or at least essentially match) the comparative models across all metrics.

[0173] FIG. 6 shows a table 600 containing K-Precision values for the example neural networks and the comparative models. The K-Precision values indicate a degree of faithfulness; i.e. how consistent the model's highest scoring caption is with the ground truth caption(s). This is different from top-1 retrieval ($R@1$) which measures exact match retrieval and does not evaluate the ability of the models to faithfully describe the elements in the image. Faithfulness has been used in the LLM literature to assess the propensity of the model to hallucinate as models with higher faithfulness more accurately capture the details of the ground truth while not inserting additional information (possible hallucinations). The lexical overlap metric of K-Precision measuring the proportion of tokens in the top chosen caption that appear in the ground truth tokens has been shown to correlate well with human judgement. Table 600 contains the K-Precision values on the MSCOCO dataset for all tokens (K-P), as well as K-Precision restricted to nouns and adjectives only (K-P_{na}), as these better encode the objects observed in the image. It can be seen that the example neural networks exhibit reduced hallucinations of objects (higher K-P_{na}) while also showing competitive performance to related methods when taking all tokens into account (as measured by K-P).

[0174] FIGS. 7 to 9 relate to fine-grained tasks that require precise localization, in particular open-vocabulary object detection and zero-shot semantic segmentation. More specifically, to evaluate whether the improved fine-grained understanding learned with process 200 translates to tasks requiring fine-grained localization, the trained example neural networks are used as a backbone for object detection. To this end, the OWL-ViT open-vocabulary object detector (Minderer et al., "Simple open-vocabulary object detection", European Conference on Computer Vision, 2022) is used with a ViT-B/16 backbone. After pre-training according to process 200, detection heads are added to the backbone and fine-tuned on Objects365 (Shao et al., "Objects365: A large-scale, high-quality dataset for object detection", Proceedings of the IEEE/CVF international conference on computer vision, 2019) and Visual Genome (Krishna et al., 2017) datasets following the approach in Minderer et al. (2022). The resulting model is evaluated on the large-vocabulary dataset LVIS (Gupta et al., "Visual genome: Connecting language and vision using crowdsourced dense image annotations", International journal of computer vision, 2019) which is well-suited for testing the transfer of knowledge from image-level pre-training. LVIS contains 1203 categories of objects, of which 307 "rare" categories are excluded from the training data to measure zero-shot transfer from pre-training. Moreover, detection on the 80 MSCOCO classes is also evaluated. Table 700 of FIG. 7 contains mean and standard deviation for three runs of the detection training. It can be seen that the example neural network improves over CLIP +0.9% on LVIS and MSCOCO as measured by mean average

precision and +3.1% on LVIS “rare” classes. Since LVIS “rare” classes are never seen during detection training data, the model has to rely on information transfer from the pre-trained representations for these classes. The large improvement of the example neural network over the comparative model on LVIS AP_{rare} suggests that the example neural network has learned more informative fine-grained representations.

[0175] FIGS. 8 and 9 relate to semantic segmentation; in particular, to zero-shot segmentation given a text label, i.e. patch embeddings are computed for a given image, and the cosine similarity of the patch embedding with the text embeddings of all the ground-truth classes is computed. A matching class is assigned for each patch as the text that corresponds to the maximum cosine similarity of that patch. Then the patches are upsampled to match the resolution of the ground-truth segmentation and for each class the Intersection over Union (IoU) between the predicted and ground-truth segmentations is calculated. The mean of the IoU scores over the classes present in the ground-truth image is calculated. Table 800 of FIG. 8 contains mIoU scores of predicted and ground-truth segmentation. It can be seen that the example neural network strongly improves over other the comparative models, significantly surpassing the next best model by +4.34 mIoU on the PASCAL VOC (Everingham et al., “The pascal visual object classes challenge: A retrospective”, International Journal of Computer Vision, 2015) dataset and by +1.2 mIoU on the PASCAL Context (Mottaghi et al., “The role of context for object detection and semantic segmentation in the wild”, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014) dataset. FIG. 9 illustrates example predicted segmentation masks on the PASCAL VOC dataset. It can be seen that CLIP predicts the object to be present in many different parts of the image, the example neural network achieves better object localization and predicts object shapes more accurately.

[0176] As noted above, the trained visual encoder neural network can be used as the neural network backbone for a subsequent task, i.e. it can be used to generate a representation of the image input that is then further processed to generate the desired output. Sometimes vision backbones trained contrastively from image-text paired data are frozen and used in foundational vision-language models (VLMs) such as Flamingo (Alayrac et al., “Flamingo: a visual language model for few-shot learning”, Advances in Neural Information Processing Systems, 2022). To demonstrate that performance improvements obtained from training a vision encoder neural network using above described process 200 can result in an improved captioning performance in VLMs, the performances of a CLIP backbone and a SPARC backbone are compared in a Flamingo-style architecture. More specifically, the ViT-B/16 vision models trained with CLIP and with the process 200 (“SPARC”) are frozen and paired

with a frozen 400M parameter (pretrained) language model. On top of the frozen vision and language backbones, Perceiver Resampler cross-attention layers are trained to produce free-form text as output. That is, the Perceiver Resampler part of Flamingo is trained on the ALIGN, LTIP (Long Text & Image Pairs) and VTP (Video & Text Pairs) datasets. VTP consists of 27 million short videos paired with text descriptions, where each video is 22s on average. The AdamW optimizer is used with a cosine learning rate schedule with peak learning rate of $1e-4$, linear warmup with 5000 warm-up steps and 250k training steps in total. The models are evaluated on captioning tasks on MSCOCO and Flickr30k datasets. Table 1000 of FIG. 10 contains CIDEr score evaluating the captioning performances. It can be seen that the SPARC backbone outperforms the CLIP backbone.

[0177] In general, FIGS. 4 to 10 demonstrate improved performances of the example neural networks compared to the comparative models over both coarse-grained tasks (like classification and retrieval) and fine-grained tasks (object detection and semantic segmentation) across a number of different benchmarks.

[0178] In this specification, the term "configured" is used in relation to computing systems and environments, as well as computer program components. A computing system or environment is considered "configured" to perform specific operations or actions when it possesses the necessary software, firmware, hardware, or a combination thereof, enabling it to carry out those operations or actions during operation. For instance, configuring a system might involve installing a software library with specific algorithms, updating firmware with new instructions for handling data, or adding a hardware component for enhanced processing capabilities. Similarly, one or more computer programs are "configured" to perform particular operations or actions when they contain instructions that, upon execution by a computing device or hardware, cause the device to perform those intended operations or actions.

[0179] The embodiments and functional operations described in this specification can be implemented in various forms, including digital electronic circuitry, software, firmware, computer hardware (encompassing the disclosed structures and their structural equivalents), or any combination thereof. The subject matter can be realized as one or more computer programs, essentially modules of computer program instructions encoded on a tangible non-transitory storage medium for execution by or to control the operation of a computing device or hardware. The storage medium can be a storage device such as a hard drive or solid-state drive (SSD), a storage medium, a random or serial access memory device, or a combination of these. Additionally or alternatively, the program instructions can be encoded on a transmitted signal, such as a machine-generated electrical, optical, or electromagnetic signal, designed to carry

information for transmission to a receiving device or system for execution by a computing device or hardware. Furthermore, implementations may leverage emerging technologies like quantum computing or neuromorphic computing for specific applications, and may be deployed in distributed or cloud-based environments where components reside on different machines or within a cloud infrastructure.

[0180] The term "computing device or hardware" refers to the physical components involved in data processing and encompasses all types of devices and machines used for this purpose. Examples include processors or processing units, computers, multiple processors or computers working together, graphics processing units (GPUs), tensor processing units (TPUs), and specialized processing hardware such as field-programmable gate arrays (FPGAs) or application-specific integrated circuits (ASICs). In addition to hardware, a computing device or hardware may also include code that creates an execution environment for computer programs. This code can take the form of processor firmware, a protocol stack, a database management system, an operating system, or a combination of these elements. Embodiments may particularly benefit from utilizing the parallel processing capabilities of GPUs, in a General-Purpose computing on Graphics Processing Units (GPGPU) context, where code specifically designed for GPU execution, often called kernels or shaders, is employed. Similarly, TPUs excel at running optimized tensor operations crucial for many machine learning algorithms. By leveraging these accelerators and their specialized programming models, the system can achieve significant speedups and efficiency gains for tasks involving artificial intelligence and machine learning, particularly in areas such as computer vision, natural language processing, and robotics.

[0181] A computer program, also referred to as software, an application, a module, a script, code, or simply a program, can be written in any programming language, including compiled or interpreted languages, and declarative or procedural languages. It can be deployed in various forms, such as a standalone program, a module, a component, a subroutine, or any other unit suitable for use within a computing environment. A program may or may not correspond to a single file in a file system and can be stored in various ways. This includes being embedded within a file containing other programs or data (e.g., scripts within a markup language document), residing in a dedicated file, or distributed across multiple coordinated files (e.g., files storing modules, subprograms, or code segments). A computer program can be executed on a single computer or across multiple computers, whether located at a single site or distributed across multiple sites and interconnected through a data communication network. The specific implementation of the computer programs may involve a combination of

traditional programming languages and specialized languages or libraries designed for GPGPU programming or TPU utilization, depending on the chosen hardware platform and desired performance characteristics.

[0182] In this specification, the term "engine" broadly refers to a software-based system, subsystem, or process designed to perform one or more specific functions. An engine is typically implemented as one or more software modules or components installed on one or more computers, which can be located at a single site or distributed across multiple locations. In some instances, one or more dedicated computers may be used for a particular engine, while in other cases, multiple engines may operate concurrently on the same one or more computers. Examples of engine functions within the context of AI and machine learning could include data pre-processing and cleaning, feature engineering and extraction, model training and optimization, inference and prediction generation, and post-processing of results. The specific design and implementation of engines will depend on the overall architecture and the distribution of computational tasks across various hardware components, including CPUs, GPUs, TPUs, and other specialized processors.

[0183] The processes and logic flows described in this specification can be executed by one or more programmable computers running one or more computer programs to perform functions by operating on input data and generating output. Additionally, graphics processing units (GPUs) and tensor processing units (TPUs) can be utilized to enable concurrent execution of aspects of these processes and logic flows, significantly accelerating performance. This approach offers significant advantages for computationally intensive tasks often found in AI and machine learning applications, such as matrix multiplications, convolutions, and other operations that exhibit a high degree of parallelism. By leveraging the parallel processing capabilities of GPUs and TPUs, significant speedups and efficiency gains compared to relying solely on CPUs can be achieved. Alternatively or in combination with programmable computers and specialized processors, these processes and logic flows can also be implemented using specialized processing hardware, such as field-programmable gate arrays (FPGAs) or application-specific integrated circuits (ASICs), for even greater performance or energy efficiency in specific use cases.

[0184] Computers capable of executing a computer program can be based on general-purpose microprocessors, special-purpose microprocessors, or a combination of both. They can also utilize any other type of central processing unit (CPU). Additionally, graphics processing units (GPUs), tensor processing units (TPUs), and other machine learning accelerators can be employed to enhance performance, particularly for tasks involving artificial intelligence and

machine learning. These accelerators often work in conjunction with CPUs, handling specialized computations while the CPU manages overall system operations and other tasks. Typically, a CPU receives instructions and data from read-only memory (ROM), random access memory (RAM), or both. The essential elements of a computer include a CPU for executing instructions and one or more memory devices for storing instructions and data. The specific configuration of processing units and memory will depend on factors like the complexity of the AI model, the volume of data being processed, and the desired performance and latency requirements. Embodiments can be implemented on a wide range of computing platforms, from small embedded devices with limited resources to large-scale data center systems with high-performance computing capabilities. The system may include storage devices like hard drives, SSDs, or flash memory for persistent data storage.

[0185] Computer-readable media suitable for storing computer program instructions and data encompass all forms of non-volatile memory, media, and memory devices. Examples include semiconductor memory devices such as read-only memory (ROM), solid-state drives (SSDs), and flash memory devices; hard disk drives (HDDs); optical media; and optical discs such as CDs, DVDs, and Blu-ray discs. The specific type of computer-readable media used will depend on factors such as the size of the data, access speed requirements, cost considerations, and the desired level of portability or permanence.

[0186] To facilitate user interaction, embodiments of the subject matter described in this specification can be implemented on a computing device equipped with a display device, such as a liquid crystal display (LCD) or an organic light-emitting diode (OLED) display, for presenting information to the user. Input can be provided by the user through various means, including a keyboard), touchscreens, voice commands, gesture recognition, or other input modalities depending on the specific device and application. Additional input methods can include acoustic, speech, or tactile input, while feedback to the user can take the form of visual, auditory, or tactile feedback. Furthermore, computers can interact with users by exchanging documents with a user's device or application. This can involve sending web content or data in response to requests or sending and receiving text messages or other forms of messages through mobile devices or messaging platforms. The selection of input and output modalities will depend on the specific application and the desired form of user interaction.

[0187] Machine learning models can be implemented and deployed using machine learning frameworks, such as TensorFlow or JAX. These frameworks offer comprehensive tools and libraries that facilitate the development, training, and deployment of machine learning models.

[0188] Embodiments of the subject matter described in this specification can be implemented within a computing system comprising one or more components, depending on the specific application and requirements. These may include a back-end component, such as a back-end server or cloud-based infrastructure; an optional middleware component, such as a middleware server or application programming interface (API), to facilitate communication and data exchange; and a front-end component, such as a client device with a user interface, a web browser, or an app, through which a user can interact with the implemented subject matter. For instance, the described functionality could be implemented solely on a client device (e.g., for on-device machine learning) or deployed as a combination of front-end and back-end components for more complex applications. These components, when present, can be interconnected using any form or medium of digital data communication, such as a communication network like a local area network (LAN) or a wide area network (WAN) including the Internet. The specific system architecture and choice of components will depend on factors such as the scale of the application, the need for real-time processing, data security requirements, and the desired user experience.

[0189] The computing system can include clients and servers that may be geographically separated and interact through a communication network. The specific type of network, such as a local area network (LAN), a wide area network (WAN), or the Internet, will depend on the reach and scale of the application. The client-server relationship is established through computer programs running on the respective computers and designed to communicate with each other using appropriate protocols. These protocols may include HTTP, TCP/IP, or other specialized protocols depending on the nature of the data being exchanged and the security requirements of the system. In certain embodiments, a server transmits data or instructions to a user's device, such as a computer, smartphone, or tablet, acting as a client. The client device can then process the received information, display results to the user, and potentially send data or feedback back to the server for further processing or storage. This allows for dynamic interactions between the user and the system, enabling a wide range of applications and functionalities.

[0190] While this specification contains many specific implementation details, these should not be construed as limitations on the scope of any invention or on the scope of what may be claimed, but rather as descriptions of features that may be specific to particular embodiments of particular inventions. Certain features that are described in this specification in the context of separate embodiments can also be implemented in combination in a single embodiment. Conversely, various features that are described in the context of a single embodiment can also

be implemented in multiple embodiments separately or in any suitable subcombination. Moreover, although features may be described above as acting in certain combinations and even initially be claimed as such, one or more features from a claimed combination can in some cases be excised from the combination, and the claimed combination may be directed to a subcombination or variation of a subcombination.

[0191] Similarly, while operations are depicted in the drawings and recited in the claims in a particular order, this should not be understood as requiring that such operations be performed in the particular order shown or in sequential order, or that all illustrated operations be performed, to achieve desirable results. In certain circumstances, multitasking and parallel processing may be advantageous. Moreover, the separation of various system modules and components in the embodiments described above should not be understood as requiring such separation in all embodiments, and it should be understood that the described program components and systems can generally be integrated together in a single software product or packaged into multiple software products.

[0192] Particular embodiments of the subject matter have been described. Other embodiments are within the scope of the following claims. For example, the actions recited in the claims can be performed in a different order and still achieve desirable results. As one example, the processes depicted in the accompanying figures do not necessarily require the particular order shown, or sequential order, to achieve desirable results. In some cases, multitasking and parallel processing may be advantageous.

CLAIMS

1. A computer-implemented method of training a neural network system comprising a visual encoder neural network and a text encoder neural network, each having a plurality of parameters; the method comprising:
 - obtaining a dataset of training data comprising a plurality of training data items, each training data item comprising an image and associated text, wherein the associated text defines a sequence of text tokens from a vocabulary of tokens;
 - at each of a plurality of training steps processing at least one of the training data items by:
 - processing pixels of the image in the training data item using the visual encoder neural network to generate a set of patch embeddings for the image, each patch embedding representing values of the pixels defining the image content of a corresponding patch of the image;
 - processing the sequence of text tokens using the text encoder neural network to generate a sequence of token embeddings representing the sequence of text tokens;
 - processing the set of patch embeddings and the sequence of token embeddings to generate a set of language-aware patch embeddings, wherein each language-aware patch embedding is determined from the set of patch embeddings based on similarities between patch embeddings of the set of patch embeddings and token embeddings of the sequence of token embeddings;
 - training at least the visual encoder neural network by backpropagating gradients of a contrastive objective function evaluated over the language-aware patch embeddings and the sequence of token embeddings, to adjust the parameters of the visual encoder neural network.
2. The method of claim 1, wherein the number of language-aware patch embeddings matches a length of the sequence of token embeddings, and, wherein the contrastive objective function is dependent on a sum, over each token embedding of the sequence of token embeddings, of an objective function having a first term dependent on a similarity between the token embedding and a respective one of the language-aware patch embeddings.
3. The method of claim 2, wherein the objective function has a second term dependent on the similarity between i) the token embedding and each language-aware patch embedding other than the respective one of the language-aware patch embeddings, or ii) the respective

one of the language-aware patch embeddings and each other token embedding than in the first term.

4. The method of any one of claims 1-3, wherein generating the set of language-aware patch embeddings comprises:

determining a similarity matrix representing the similarities between patch embeddings of the set of patch embeddings and token embeddings of the sequence of token embeddings, each element of the similarity matrix representing a similarity between one of the patch embeddings and one of the token embeddings; and

generating the set of language-aware patch embeddings from the set of patch embeddings based on the similarity matrix.

5. The method of claim 4, further comprising:

processing the similarity matrix to set a subset of the elements of the similarity matrix to zero.

6. The method of claim 5, further comprising processing the image in the training data item to select a portion of the image; and using the selected portion of the image to select the subset of the elements of the similarity matrix to set to zero.

7. The method of any one of claims 4-6, further comprising:

processing the similarity matrix to set to zero elements of the similarity matrix for which the similarity between the patch embedding and the token embedding corresponding to the element does not meet a similarity criterion.

8. The method of claim 7, wherein the similarity between a patch embedding and a token embedding is defined by a metric of similarity between the patch embedding and the token embedding; the method comprising:

processing the similarity matrix to set to zero elements of the similarity matrix for which the metric of similarity does not satisfy a threshold value.

9. The method of claim 8, wherein processing the similarity matrix to set to zero elements of the similarity matrix for which the metric of similarity does not satisfy a threshold value comprises:

normalizing the elements of the similarity matrix; and

for each element of the similarity matrix, comparing the value of the element to the threshold value and setting the value to zero when the value of the element is less than the threshold value.

10. The method of claim 8 or 9, wherein the threshold value is dependent on $1/P$ where P is the number of patch embeddings.

11. The method of any one of claims 4-10, wherein generating the set of language-aware patch embeddings from the set of patch embeddings based on the similarity matrix comprises:

generating each language-aware patch embedding by, for each patch embedding determining a set of attention weights, one attention weight for each of the token embeddings, from the elements of the similarity matrix for that patch embedding and for each of the token embeddings; and

determining a language-aware patch embedding, for each token embedding, by:

combining each of the patch embeddings according to the respective weight for that patch embedding and for the token embedding.

12. The method of any one of claims 1-11, wherein generating the set of language-aware patch embeddings comprises:

determining, for each patch embedding of the set of patch embeddings, an attention weight for each token embedding, by:

for each patch embedding and for each token embedding of the sequence of token embeddings, determining the attention weight for the patch embedding and for the token embedding from a metric of similarity between the patch embedding and the token embedding; and

determining, for each token embedding, one the language-aware patch embeddings by:

combining the patch embeddings according to the attention weights for the token embedding and for the respective patch embeddings.

13. The method of any one of claims 1-12, further comprising, at each of the training steps, processing a batch of the training data items by:

for each training data item in the batch of training data items:
determining a global image embedding for the image in the training data item by combining the set of patch embeddings for the image, and
determining a global text embedding for the associated text in the training data item by combining the text tokens in the sequence of text tokens for the associated text;
evaluating a global contrastive objective for the batch of training data items dependent on a sum, for each training data item, of a global contrastive objective function having a first term dependent on a similarity between the global image embedding for the training data item and the global text embedding for the training data item; and
training the visual encoder neural network and text encoder neural network by backpropagating gradients of the global contrastive objective function.

14. The method of claim 13, wherein the global objective function has a second term dependent on i) the similarity between the global image embedding for the training data item and the global text embedding for each of the other training data items in the batch, or ii) the similarity between the global text embedding for the training data item and the global image embedding for each of the other training data items in the batch.

15. The method of any preceding claim, comprising evaluating the contrastive objective function over the language-aware patch embeddings and the sequence of token embeddings from only a single training data item.

16. The method of any preceding claim, comprising jointly training both the visual encoder neural network and the text encoder neural network by backpropagating gradients of the contrastive objective function to adjust the parameters of both the visual encoder neural network and the text encoder neural network.

17. A computer-implemented method of performing an image processing task, the method comprising:

receiving one or more patch embeddings and/or a global image embedding generated using a visual encoder neural network trained according to any one of claims 1-16, and using the one or more patch embeddings and/or the global image embedding to perform an image processing task, wherein performing the image processing task comprises:

processing pixels of an image to generate the set of patch embeddings and/or global image embedding for the image; and

using the set of patch embeddings and/or global image embedding to perform the image processing task, wherein the image processing task comprises:

processing the image to provide output data that identifies the presence or location of one or more objects in the image; or

processing the image to provide output data that segments pixels of the image into regions that represent one or more objects in the image; or

processing the image to provide output data that categorizes a content of the image into one or more of a plurality of categories; or

processing the image to provide output data that predicts depth values for pixels of the image; or

processing the image to provide output data that is used to control an action of a mechanical agent acting in a real-world environment to perform a task; or

when the image comprises a moving image the image processing task comprises:

processing the image to provide output data that identifies the location of one or more actions represented in the video; or

processing the image to provide output data that categorizes one or more actions represented in the video into one or more of a plurality of categories; or

processing the image to provide output data that predicts one or more events that involve objects in the image; or,

when the method further comprises receiving a sequence of token embeddings from a trained text encoder neural network, processing the one or more patch embeddings and/or the global image embedding and the sequence of token embeddings to perform an image processing task defined by a sequence of text tokens encoded by the sequence of token embeddings.

18. A computer-implemented method of performing an image processing task, the method comprising:

providing an image to a visual encoder neural network, the visual encoder neural network having been trained by performing the respective operations of the method of any one of claims 1-16;

processing pixels of the image using the visual encoder neural network to generate the set of patch embeddings or the global image embedding for the image; and

processing the set of patch embeddings or the global image embedding for the image to perform the image processing task, wherein the image processing task comprises:

processing the image to provide output data that identifies the presence or location of one or more objects in the image; or

processing the image to provide output data that segments pixels of the image into regions that represent one or more objects in the image; or

processing the image to provide output data that categorizes a content of the image into one or more of a plurality of categories; or

processing the image to provide output data that predicts depth values for pixels of the image; or

processing the image to provide output data that is used to control an action of a mechanical agent acting in a real-world environment to perform a task; or

when the image comprises a moving image the image processing task comprises:

processing the image to provide output data that identifies the location of one or more actions represented in the video; or

processing the image to provide output data that categorizes one or more actions represented in the video into one or more of a plurality of categories; or

processing the image to provide output data that predicts one or more events that involve objects in the image; or

when the method further comprises receiving a sequence of token embeddings from a trained text encoder neural network, processing the one or more patch embeddings and/or the global image embedding and the sequence of token embeddings to perform an image processing task defined by a sequence of text tokens encoded by the sequence of token embeddings.

19. A computer-implemented method of processing, using a neural network system comprising a visual encoder neural network and a text encoder neural network, an image and associated text to generate a set of language-aware patch embeddings; the method comprising:

obtaining the image and the associated text, wherein the associated text defines a sequence of text tokens from a vocabulary of tokens;

processing pixels of the image using the visual encoder neural network to generate a set of patch embeddings for the image, each patch embedding representing values of the pixels defining the image content of a corresponding patch of the image;

processing the sequence of text tokens using the text encoder neural network to generate a sequence of token embeddings representing the sequence of text tokens; and

processing the set of patch embeddings and the sequence of token embeddings to generate the set of language-aware patch embeddings, wherein each language-aware patch embedding is determined from the set of patch embeddings based on similarities between patch embeddings of the set of patch embeddings and token embeddings of the sequence of token embeddings.

20. A system comprising:

one or more computers; and

one or more storage devices communicatively coupled to the one or more computers, wherein the one or more storage devices store instructions that, when executed by the one or more computers, cause the one or more computers to perform the operations of the respective method of any one of claims 1-19.

21. One or more non-transitory computer storage media storing instructions that when executed by one or more computers cause the one or more computers to perform the operations of the respective method of any one of claims 1-19.

22. A visual encoder neural network comprising a plurality of trainable parameters and configured to process pixels of an image to generate a set of patch embeddings for the image, each patch embedding representing values of the pixels defining the image content of a corresponding patch of the image, wherein the patch embeddings and corresponding text token embeddings representing a textual description of the image have a similarity relationship, according to a similarity metric, such that patch embeddings corresponding to image regions described by a portion of the textual description have a higher similarity to the text token embedding representing said portion of the textual description than to text token embeddings representing other portions of the textual description.

23. The visual encoder neural network of claim 22, further configured to generate a global image embedding for the image by combining the set of patch embeddings.

24. The visual encoder neural network of claim 23, wherein the global image embedding is generated by average pooling the set of patch embeddings, optionally followed by processing the combined patch embeddings using one or more non-linear layers.

25. The visual encoder neural network of any of claims 23 or 24, wherein the global image embedding has a higher similarity, according to the similarity metric, to a global text embedding representing a caption of the image than to global text embeddings representing other texts.

26. The visual encoder neural network of any one of claims 22 to 25, wherein the similarity metric is selected from the group consisting of cosine similarity, dot product similarity, and Euclidean distance.

27. The visual encoder neural network of any one of claims 22 to 26, wherein the visual encoder neural network has been trained using a dataset of image-text pairs and a contrastive loss function that encourages alignment between the patch embeddings and corresponding text token embeddings.

28. A computer-implemented method of performing an image processing task, the method comprising:

receiving one or more patch embeddings for an image, each patch embedding representing values of pixels defining the image content of a corresponding patch of the image, wherein the patch embeddings and corresponding text token embeddings representing a textual description of the image have a similarity relationship, according to a similarity metric, such that patch embeddings corresponding to image regions described by a portion of the textual description have a higher similarity to the text token embedding representing said portion of the textual description than to text token embeddings representing other portions of the textual description; and

using the one or more patch embeddings to perform an image processing task.

29. A set of patch embeddings for an image, each patch embedding representing values of the pixels defining the image content of a corresponding patch of the image, wherein the patch embeddings and corresponding text token embeddings representing a textual description of the image have a similarity relationship, according to a similarity metric, such

that patch embeddings corresponding to image regions described by a portion of the textual description have a higher similarity to the text token embedding representing said portion of the textual description than to text token embeddings representing other portions of the textual description.

30. The set of patch embeddings of claims 29, wherein the similarity metric is selected from the group consisting of cosine similarity, dot product similarity, and Euclidean distance.

1/9

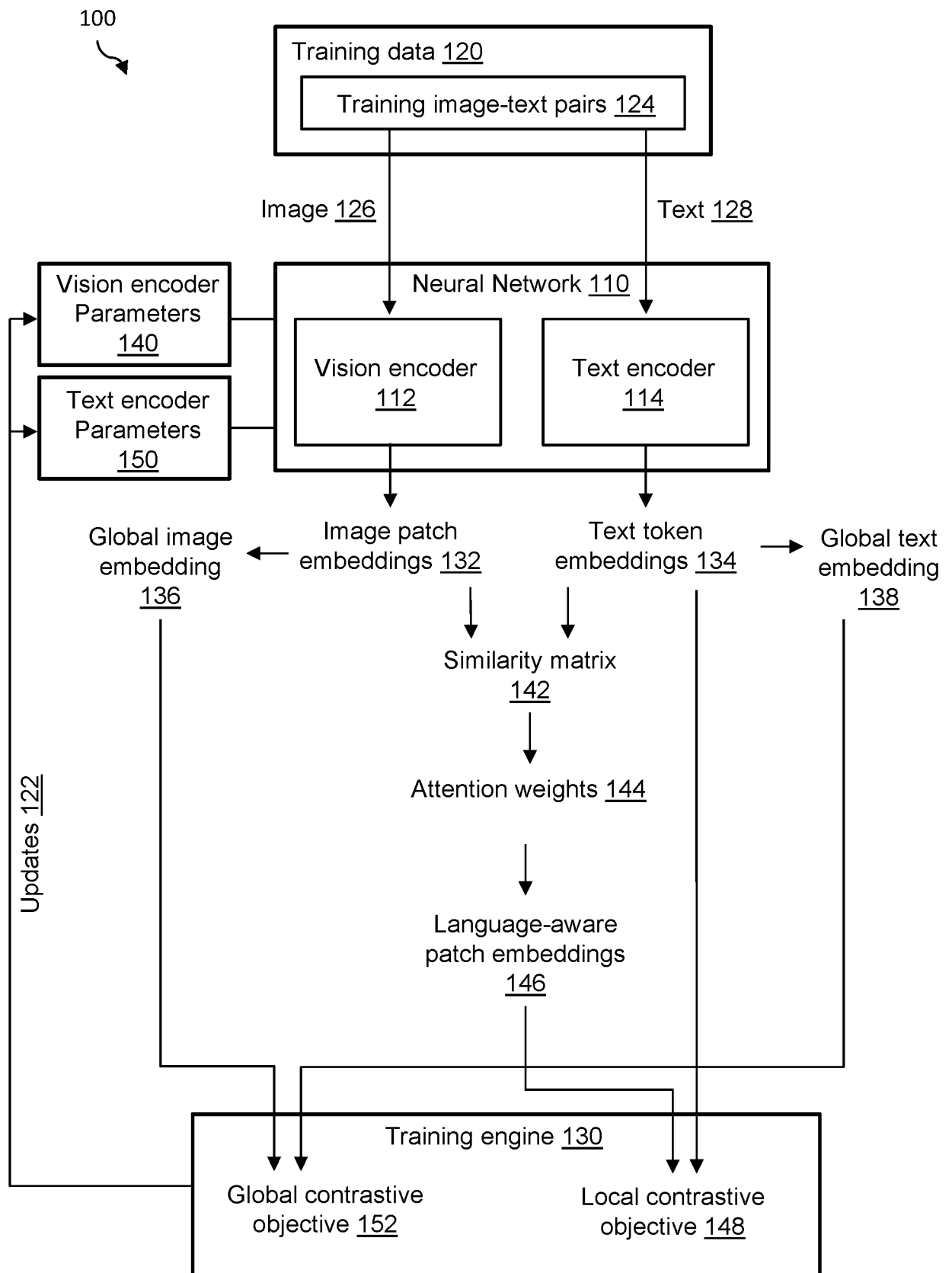


Figure 1

2/9

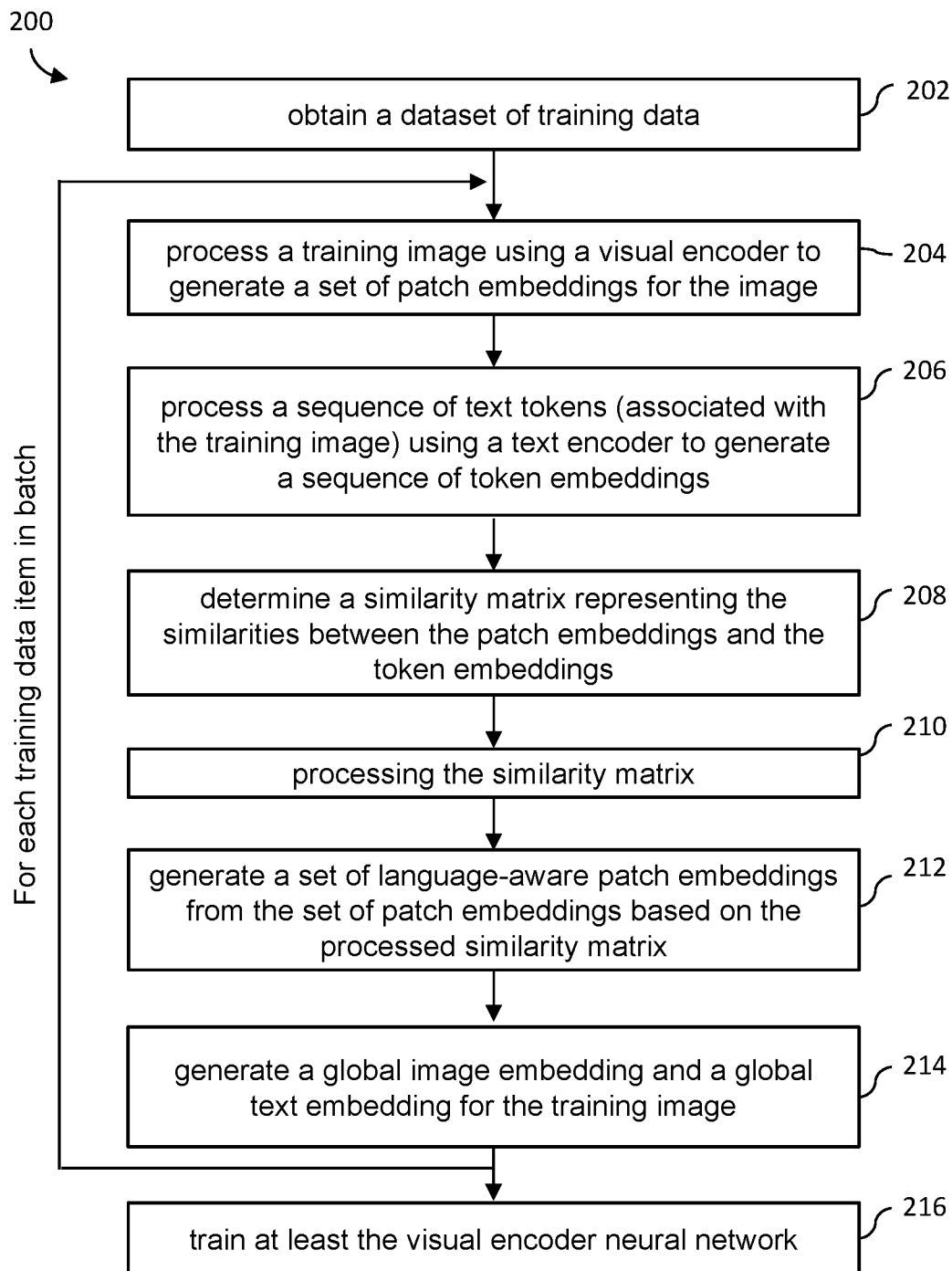


Figure 2

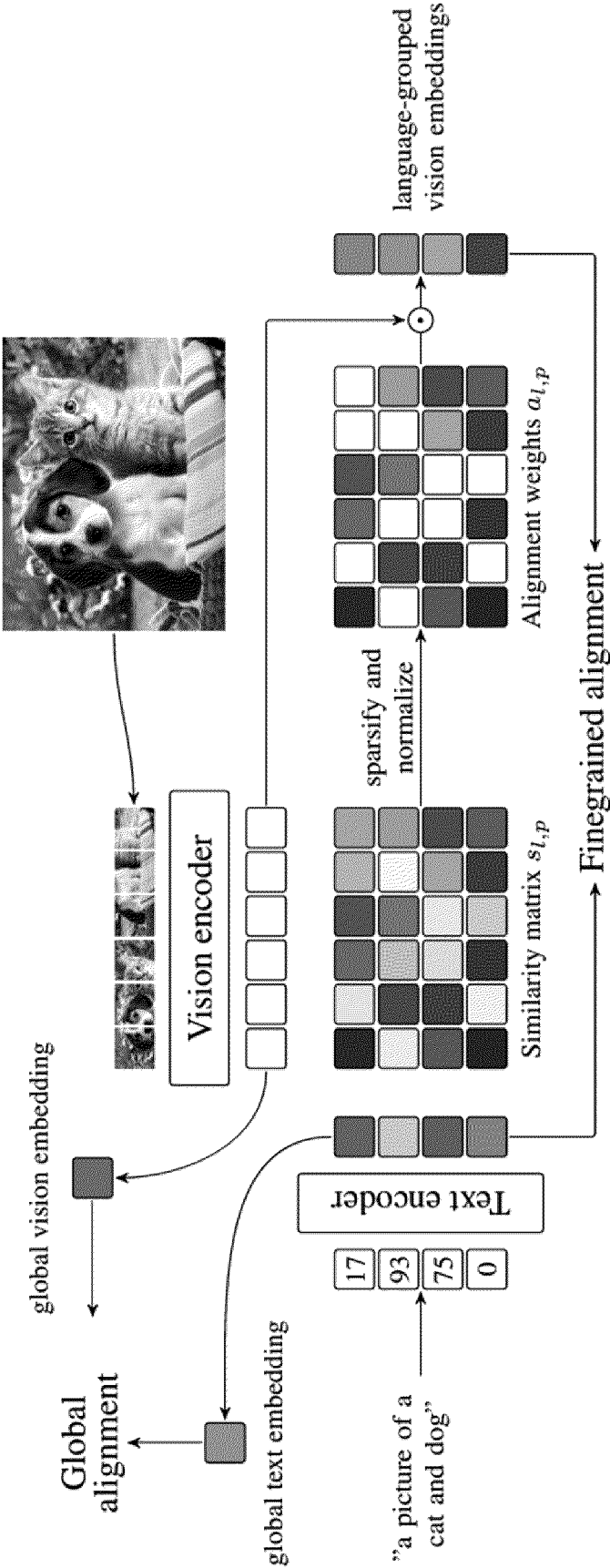


Figure 3

400

	Objective	IN	IN-V2 Th	IN-V2 MF	IN-V2 TI	IN-R	IN-C	IN-A	IN-Sketch
ViT-B/32	CLIP	66.7	66.2	58.9	71.5	63.2	42.6	15.1	51.7
	FILIP	52.7	50.7	44.0	55.8	47.1	28.7	8.4	38.2
	PACL	58.9	56.9	50.0	62.6	54.0	34.9	9.3	44.1
	GloRIA	62.8	61.5	54.3	66.7	56.7	38.4	11.2	47.5
	MGCA	66.0	64.5	56.4	69.5	62.0	41.1	14.7	51.7
	SPARC (ours)	68.1	67.0	59.7	72.0	64.9	44.5	16.7	53.2
ViT-B/16	CLIP	71.6	70.9	63.7	74.8	71.1	48.5	32.2	56.8
	FILIP	56.6	55.6	48.9	59.7	54.0	33.2	14.4	43.1
	PACL	61.1	59.6	52.6	64.8	56.3	36.1	12.8	45.2
	GloRIA	67.4	66.9	59.8	71.7	66.6	43.8	24.6	54.2
	MGCA	69.6	69.3	62.2	73.6	68.8	46.1	29.0	55.0
	SPARC (ours)	72.6	71.1	64.4	75.0	72.0	48.5	33.8	57.3
ViT-L/4	CLIP	77.3	75.9	69.5	79.1	78.8	59.6	52.5	64.5
	MGCA	75.6	73.9	68.0	77.9	77.2	56.0	45.0	63.1
	SPARC (ours)	78.2	76.9	70.6	80.0	79.3	59.7	51.9	65.4

402

	Objective	IN	IN-V2 Th	IN-V2 MF	IN-V2 TI	IN-R	IN-C	IN-A	IN-Sketch
ViT-B/32	CLIP	69.0	68.8	60.4	73.4	62.4	44.6	15.8	52.4
	FILIP	56.8	54.8	48.4	60.0	44.6	30.8	7.8	39.6
	PACL	61.2	59.5	51.9	65.2	52.9	36.4	9.3	45.2
	GloRIA	65.9	64.8	57.0	69.6	57.4	40.7	11.7	48.7
	MGCA	68.6	67.4	59.2	72.6	61.0	43.5	14.1	50.9
	SPARC (ours)	70.4	69.6	62.1	74.5	63.2	46.5	17.3	52.7
ViT-B/16	CLIP	73.9	73.6	66.1	77.1	68.8	50.4	32.5	57.3
	FILIP	61.4	61.0	53.8	65.6	53.2	35.9	14.2	45.1
	PACL	63.3	61.7	54.4	66.8	54.1	37.3	12.9	45.4
	GloRIA	70.4	70.0	62.8	74.7	65.7	46.4	25.0	54.8
	MGCA	72.7	72.7	65.3	76.3	67.6	48.4	29.8	55.5
	SPARC (ours)	74.7	74.0	67.1	77.8	71.1	51.31	34.2	57.9
ViT-L/4	CLIP	79.2	78.5	71.8	81.6	78.5	61.3	51.5	65.1
	MGCA	78.0	77.4	70.5	80.6	75.2	57.9	45.5	63.1
	SPARC (ours)	79.7	78.9	72.6	81.9	79.8	61.3	53.4	65.9

Figure 4

500		Flickr30k				MSCOCO							
Objective		image-to-text		text-to-image		image-to-text		text-to-image					
		R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10			
ViT-B/32	CLIP	79.2	95.1	97.2	66.5	88.0	93.1	53.5	78.2	86.7	38.4	64.8	74.9
	PACL	65.5	86.8	92.2	49.8	76.5	84.7	37.6	65.1	75.7	26.5	50.6	61.8
	GLoRIA	74.6	92.1	96.2	61.5	85.3	90.7	46.9	73.0	82.7	34.5	61.0	71.7
	MGCA	81.5	93.9	96.8	64.4	86.5	92.0	54.5	78.6	86.8	37.7	63.7	74.0
	FILIP	62.6	86.9	92.9	50.5	77.7	84.9	35.6	61.0	73.1	26.2	51.0	62.4
SPARC (ours)		82.5	96.2	97.6	67.7	88.2	93.0	55.0	79.1	87.3	39.7	65.9	75.7
ViT-B/16	CLIP	84.0	96.1	98.2	71.6	90.3	94.1	56.2	80.6	88.2	42.4	68.6	78.3
	PACL	69.6	89.7	94.2	54.9	80.7	87.3	41.8	67.8	77.6	29.1	54.3	65.5
	GLoRIA	78.0	95.5	98.0	68.4	88.9	93.2	49.7	75.4	84.6	38.9	65.1	75.2
	MGCA	82.2	96.1	98.1	67.7	88.5	93.2	57.6	80.5	87.8	39.8	65.7	75.3
	FILIP	69.0	89.8	94.0	55.8	81.5	87.9	40.2	66.0	76.3	29.5	55.3	66.3
SPARC (ours)		84.4	97.6	98.7	72.0	91.2	94.9	57.6	81.2	88.5	43.0	68.6	78.5
ViT-L/14	CLIP	84.7	96.9	98.4	73.7	91.8	95.4	58.6	82.6	89.1	44.8	70.5	79.5
	MGCA	85.9	96.9	98.1	73.2	91.6	95.3	59.7	83.2	89.7	44.3	69.6	78.8
	SPARC (ours)	86.9	97.3	98.6	74.4	91.7	95.4	58.9	82.9	89.7	45.6	71.1	80.1

Figure 5

600

Method	ViT-B/32		ViT-B/16	
	$\mathcal{K}\text{-P}_{\text{na}}$	$\mathcal{K}\text{-P}$	$\mathcal{K}\text{-P}_{\text{na}}$	$\mathcal{K}\text{-P}$
CLIP	76.03	77.82	77.56	78.99
FILIP	63.3	66.83	66.05	70.09
PACL	3.36	26.26	4.09	27.31
GLoRIA	71.63	73.54	73.85	75.3
MGCA	75.79	77.98	77.66	80.03
SPARC (ours)	76.46	78.44	78.72	79.77

Figure 6

700
/

Method	LVIS		MSCOCO
	AP _{all}	AP _{rare}	AP _{all}
CLIP	26.9 ± 0.12	22.0 ± 0.79	38.5 ± 0.19
SPARC (ours)	27.9 ± 0.11	25.1 ± 0.95	39.4 ± 0.13

Figure 7

800
/

Method	Pascal VOC	Pascal Context
CLIP	23.02	20.45
FILIP	19.32	9.31
PACL	1.23	1.61
GLoRIA	22.64	15.26
MGCA	21.91	11.50
SPARC (ours)	27.36	21.65

Figure 8

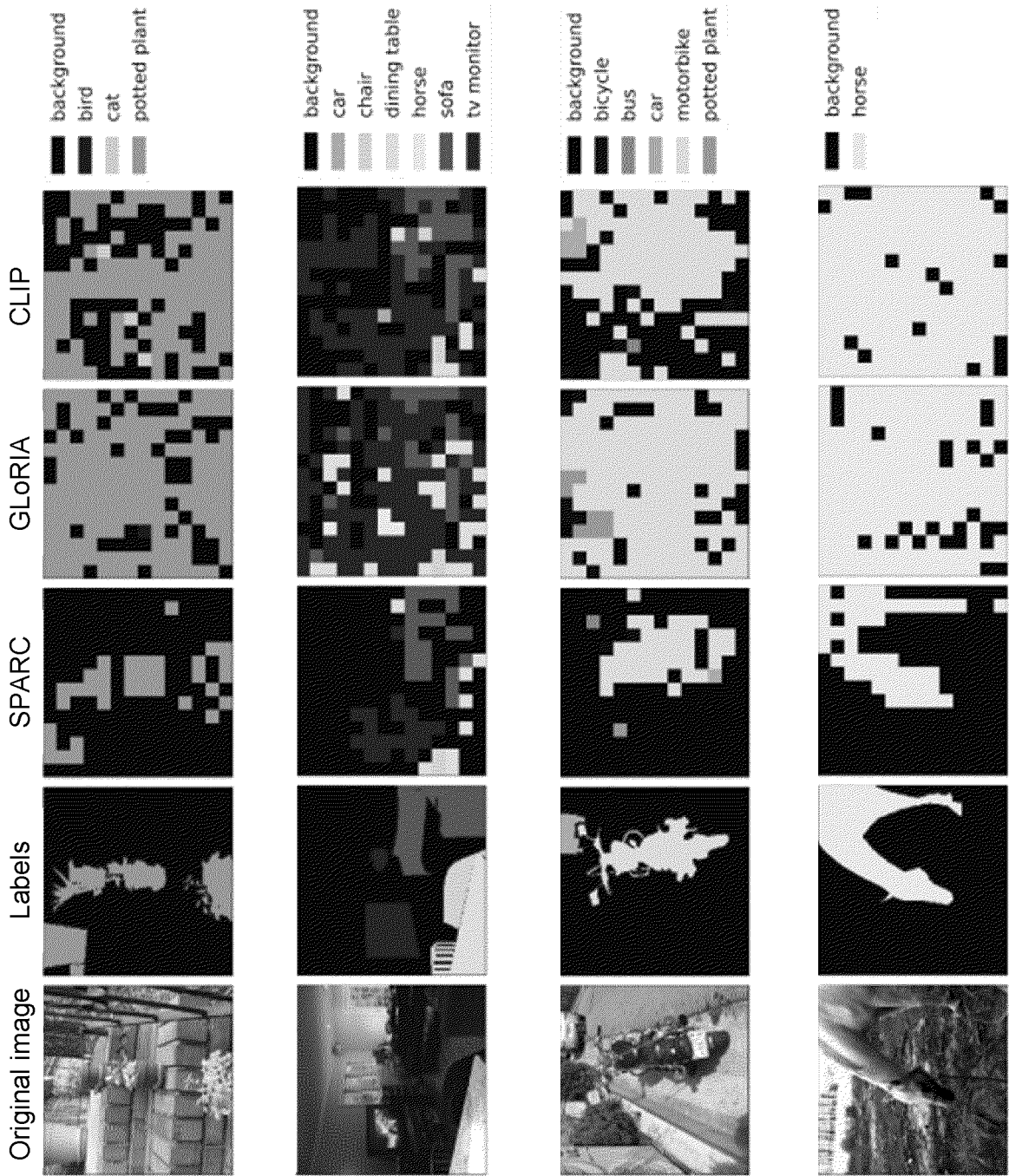


Figure 9

1000
/

Method	MSCOCO	Flickr30k
CLIP	24.3	12.9
SPARC (ours)	25.3	13.6

Figure 10

INTERNATIONAL SEARCH REPORT

International application No

PCT/EP2024/082601

A. CLASSIFICATION OF SUBJECT MATTER

INV. G06N3/045 G06N3/084 G06N3/09
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06N

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-Internal**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	HUANG SHIH-CHENG ET AL: "GLORIA: A Multimodal Global-Local Representation Learning Framework for Label-efficient Medical Image Recognition", 2021 IEEE/CVF INTERNATIONAL CONFERENCE ON COMPUTER VISION (ICCV), 1 October 2021 (2021-10-01), pages 3922-3931, XP055967144, DOI: 10.1109/ICCV48922.2021.00391 ISBN: 978-1-6654-2812-5 Retrieved from the Internet: URL:https://ieeexplore.ieee.org/stampPDF/getPDF.jsp?tp=&arnumber=9710099&ref=aHR0cHM6Ly9pZWVleHBsb3JlLml1ZWUub3JnL2Fic3RyYWN0L2RvY3VtZW50Lzk3MTAwOTk=>abstract chapters 1-5; page 3922 - page 3929; figures 1-4 ----- - / - -	1 - 30



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

4 February 2025

Date of mailing of the international search report

12/02/2025

Name and mailing address of the ISA/

European Patent Office, P.B. 5818 Patentlaan 2

NL - 2280 HV Rijswijk

Tel. (+31-70) 340-2040,

Fax: (+31-70) 340-3016

Authorized officer

Hasnas, Sergiu

INTERNATIONAL SEARCH REPORT

International application No

PCT/EP2024/082601

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	JINGHAO ZHANG ET AL: "Latent Structures Mining with Contrastive Modality Fusion for Multimedia Recommendation", ARXIV.ORG, CORNELL UNIVERSITY LIBRARY, 201 OLIN LIBRARY CORNELL UNIVERSITY ITHACA, NY 14853, 1 November 2021 (2021-11-01), XP091091542, abstract section 2.2.1; page 3 - page 4 -----	5 - 10