

TRANSFORMER FOR OPTICAL CHARACTER RECOGNITION

BACKGROUND

[0001] A transformer is an attention-based neural network that can be applied to sequential input data without recurrence. Transformer models can be used as a backbone network for downstream tasks, such as image classification and optical character recognition (OCR).

[0002] Conventional transformer models are based on an encoder-decoder architecture, in which one or more encoder layers process an input sequence to generate an encoder representation of the input sequence and one or more decoder layers process the encoder representation to generate the transformer output. The one or more decoder layers autoregressively output predicted results in a sequential manner. As a result, it can be challenging to generate predictions in parallel. Depending on the application, this may lead to excessive runtime latency when processing an input image to recognize a plurality of words or word pieces.

SUMMARY

[0003] To address the issues discussed above, a computing system is provided comprising a processor and a memory storing instructions executable by the processor. The instructions are executable to obtain an input image including text for optical character recognition. The input image is decomposed into a plurality of patches. The plurality of patches are arranged in an input sequence. A patch embedding is generated for each of the plurality of patches via a patch embedding layer. The patch embedding includes a vector representation of the patch. A position embedding is generated for each of the plurality of patches via a position embedding layer. The position embedding indicates a position of the patch in the input sequence. The

position embeddings and patch embeddings for each of the plurality of input patches are provided to a transformer encoder of a neural network according to the input sequence to thereby cause the transformer encoder to output an encoder representation of the input sequence. The encoder representation is provided to a multi-class output layer of the neural network configured with a plurality of nodes. Each node represents a word or a word piece in a tokenized vocabulary. This causes the multi-class output layer to output a predicted vocabulary word or word piece for the encoder representation of the input sequence. A text representation of the text of the input image is generated including the predicted vocabulary word or word piece.

[0004] This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used to limit the scope of the claimed subject matter. Furthermore, the claimed subject matter is not limited to implementations that solve any or all disadvantages noted in any part of this disclosure.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] FIG. 1 shows a schematic diagram of an example of a computing system for generating a text representation of text contained within an input image according to an example embodiment of the subject disclosure, at runtime.

[0006] FIG. 2 shows a schematic diagram of an example of a transformer neural network that can be implemented at the computing system of FIG. 1.

[0007] FIG. 3 shows the computing system of FIG. 1 in an example of a pre-training configuration.

[0008] FIG. 4 shows the computing system of FIG. 1 in an example of a fine-tuning configuration.

[0009] FIG. 5 shows the computing system of FIG. 1 in another example of a fine-tuning configuration.

[0010] FIG. 6 is a flowchart illustrating an example method for generating a text representation of text contained within an input image according to an example embodiment of the subject disclosure.

[0011] FIG. 7 is a schematic diagram illustrating an example computing system that may be used to implement the computing system of FIG. 1.

DETAILED DESCRIPTION

[0012] As introduced above, transformer models utilize attention-based deep learning networks to process sequential input data. In some instances, transformer-based text recognition models used for optical character recognition (OCR) have an encoder-decoder architecture, in which an image transformer serves as an encoder for image understanding. A text transformer serves as a decoder to generate output text. However, conventional transformer decoders autoregressively generate predicted results (e.g., words or word pieces) in a sequential manner. As a result, transformer-based OCR systems may have greater runtime latency than systems based on other neural networks. This can make transformers difficult to deploy in real-world use cases (e.g., cloud-based OCR services).

[0013] In some instances, a transformer-based system may be used to perform OCR on a character-by-character level. This may reduce latency relative to generating predicted output text at the word or word piece level. However, character-level text prediction may be less accurate than word- or word-piece-level text prediction.

[0014] Accordingly, examples are disclosed that relate to systems and methods for generating a text representation of text contained within an input image. Briefly, a transformer encoder of a neural network is used to output an encoder representation of an input sequence. A multi-class output layer of the neural network outputs a predicted vocabulary word or word piece for the encoder representation of the input sequence. The multi-class output layer is sequence invariant. In this manner, the multi-class output layer may generate a plurality of words or word pieces in parallel. This enables the neural network to generate the text representation of the text within the input image faster than another transformer neural network that employs an autoregressive decoder.

[0015] FIG. 1 shows an example of a computing system 102 for generating a text representation 104 of text 106 contained within an input image 108. In some examples, the computing system 102 comprises a server computing system (e.g., a cloud-based server or a plurality of distributed cloud servers). In other examples, the computing system 102 may comprise any other suitable type of computing system. Other examples of suitable computing systems include, but are not limited to, a desktop computer and a laptop computer. Additional aspects of the computing system 102 are described in more detail below with reference to FIG. 7.

[0016] The computing system 102 is configured to obtain the input image 108 including text 106 for optical character recognition. In some examples, the computing system 102 is configured to obtain the input image 108 from a client 110. The computing system 102 is optionally configured to output the text representation 104 to the client 110. In some examples, the client 110 comprises a computing system separate from the computing system 102. Some examples of suitable computing systems include, but are not limited to, a desktop computing device, a laptop

computing device, or a smartphone. Additional aspects of the client 110 are described in more detail below with reference to FIG. 7.

[0017] The input image 108 is decomposed into a plurality of patches 112. In some examples, decomposing the input image 108 includes normalizing the input image 108 to a normalized image resolution, and decomposing the normalized input image into the plurality of patches 112. In this manner, the normalized input image is the same size as each training image.

[0018] For example, an input image $x_{\text{img}} \in \mathbb{R}^{3 \times H_0 \times W_0}$ is resized to a fixed size (H, W) and then decomposed into $N = HW/P^2$ patches. Here, (H, W) is the normalized input image resolution and (P, P) is the resolution of each patch, where the width W and the height H of the normalized input image are divisible by the patch size P . In some examples, the normalized input image is 224 pixels x 224 pixels, which is decomposed into a sequence of 14 x 14 patches, with each patch having a size of 16 pixels x 16 pixels. In other examples, the normalized input image and/or the patches may have any other suitable dimensions. Other examples of suitable normalized input image resolutions include, but are not limited to, 196 pixels x 196 pixels and 384 pixels x 384 pixels. Other examples of suitable patch sizes include, but are not limited to, 14 pixels x 14 pixels and 24 x 24 pixels. The use of smaller input images may reduce runtime latency by a factor of n^2 . For example, a transformer may process a 224-pixel image 50% faster than a 384-pixel image.

[0019] To process the input image 108, the plurality of patches are arranged in an input sequence. A patch embedding 114 is generated for each patch 112 of the plurality of patches 112 via a patch embedding layer 116. In some examples, the patch embedding 114 comprises a vector representation 118 of the patch 112. In some such examples, each patch 112 of the plurality of patches 112 is flattened and linearly

projected into a D-dimension vector, which serves as the patch embedding 114. Here, D is the hidden size of the transformer through all of its layers. This converts the input image into a format that can be processed by the transformer.

[0020] In some examples, a classification token 120 (e.g., [CLS]) is provided as an input to the transformer encoder along with the patch embeddings 114. The classification token 120 brings together information from the plurality of patch embeddings 114 and represents the whole input image 108. In some examples, a distillation token 122 (e.g., [DiT]) is additionally or alternatively provided to the transformer encoder. The distillation token allows the image transformer to reproduce a label predicted by a teacher model in examples where the transformer encoder is pre-trained via a student-teacher strategy.

[0021] The computing system 102 is further configured to generate a position embedding 124 for each of the plurality of patches via a position embedding layer 126. The position embeddings 124 indicate a position of each patch 112 in the input sequence. In the example depicted in FIG. 1, each of the patch embeddings 114 are given learnable 1-dimension (1D) position embeddings 1-N, where N is the number of patches 112 in the input sequence. Inclusion of the classification token 120 and the distillation token 122 results in an input sequence of length N+2. The classification token 120 and the distillation token 122 are depicted in FIG. 1 with position embeddings 124 of N+1 and N+2, respectively. However, it will also be appreciated that the classification token 120 and the distillation token 122 may have any other suitable position embeddings, such as position embeddings encoding their absolute positions in the input sequence (e.g., 1 and 42, respectively).

[0022] As introduced above, the computing device 102 comprises a transformer neural network 128. The transformer neural network 128 includes a

transformer encoder 130. In some examples, the transformer encoder 130 comprises a single image transformer layer. In other examples, the transformer neural network 128 includes a plurality of stacked transformer encoder layers. In some examples, the transformer neural network 128 includes 3-12 identical transformer encoder layers. Stacking a plurality of transformer encoder layers enables the transformer neural network 128 to extract features from input embeddings in a hierarchical manner and provides the transformer neural network with a greater number of parameters to model relationships between the input embeddings and output text than the use of fewer encoder layers. On the other hand, the use of fewer encoder layers (e.g., 1-6) results in faster training and runtime operation, but may result in less accurate inferences than the use of a greater number of encoder layers (e.g., 7 or more).

[0023] As indicated at 111, the position embeddings 124 and patch embeddings 114 for each of the plurality of input patches 112 are provided to the transformer encoder 130 of the transformer neural network 128. This causes the transformer encoder to output an encoder representation 146 of the input sequence during a run-time inference phase.

[0024] To generate the encoder representation 146, and with reference now to FIG. 2, each transformer encoder 130 includes an attention module 132. The attention module 132 enables the transformer neural network 128 to pay closer attention to a particular part of the input sequence while selectively ignoring or placing less weight on other parts. The attention module 132 is configured to output an attention-weighted sum of values, where the weights of the values are computed using a compatibility function of a query with a corresponding key. The queries, keys and values are extracted from the same input sequence. In some examples, the compatibility function computes the dot product of the query and the key. In other examples, any other

suitable attention function may be used. For example, and as described in more detail below with reference to equation (1), the transformer encoder may employ scaled dot-product attention.

[0025] (1) $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$

[0026] In equation (1), Q represents a set of queries, K^T represents a set of keys, and V represents a set of values. In equation (1), which represents a scaled dot-product attention function, the attention function includes a scaling factor $\frac{1}{\sqrt{d_k}}$. In the scaling factor, d_k is the dimension of the queries and keys. The scaling factor prevents extremely large dot products from leading the softmax function in equation (1) to regions with tiny gradients.

[0027] In some examples, the attention module 132 includes a multi-head attention function 134. Equation (2) provides one example of a multi-head attention function ($\text{MultiHead}(Q, K, V)$).

[0028] (2) $\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^O,$
 where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V).$

[0029] In equation (2), W_i^Q , W_i^K , W_i^V and W^O are projection matrices. The multi-head attention function 134 linearly projects the queries (Q), keys (K) and values (V) h -times with different, learned linear projection weights, rather than evaluating a single attention function. For each group of the projected queries, keys and values, the multi-head attention function computes attention in parallel, concatenates each head and once again projects the attention values. In this manner, the multi-head attention function enables the model to jointly attend to information from different representation subspaces at different positions.

[0030] The transformer neural network 128 further comprises one or more feed-forward layers 136. The one or more feed-forward layers 136 receive, as input, the output of the attention module 132. The one or more feed-forward layers 136 are configured to concentrate probability mass on tokens likely to correspond to patterns in the attention module output. In this manner, the one or more feed-forward layers enable the transformer encoder 130 to recognize and highlight semantic patterns in the input sequence.

[0031] In some examples, the transformer encoder 130 further includes a residual connection layer 138 and a normalization layer 140 between the attention module 132 and the one or more feed-forward layers 136. The residual connection layer 138 is configured to add the output of the attention module to an original input to the transformer encoder 130. In some such examples, the transformer encoder additionally or alternatively includes an additional residual connection layer 142 and an additional normalization layer 144 downstream of the one or more feed-forward layers 136. This provides an additional path for the original input to reach downstream layers of the transformer neural network 128.

[0032] The transformer neural network 128 also includes a multi-class output layer 148. The computing system 102 is configured to provide the encoder representation 146 to the multi-class output layer 148. The multi-class output layer 148 comprises a plurality of nodes 150. Each node 150 of the plurality of nodes 150 represents a word or a word piece 152 in a tokenized vocabulary 154. As described in more detail below, the multi-class output layer 148 outputs a predicted vocabulary word or word piece responsive to receiving the encoder representation. The predicted vocabulary word or word piece may take the form of a probability value for a

corresponding node 150. This enables the computing system 102 to generate the text representation 104.

[0033] Each node 150 of the multi-class output layer 148 includes an activation function. In some examples, each node of the multi-class output layer 148 includes a softmax function 156. In other examples, the multi-class output layer 148 may include any other suitable function. Other examples of suitable functions include, but are not limited to, sigmoid functions, linear functions, and step functions. The activation function is configured to output a probability that its corresponding vocabulary word or word piece 152 matches an input vector.

[0034] Advantageously, and in contrast to an attention-based transformer decoder, each node 150 of the multi-class output layer 148 is sequence invariant (e.g., non-autoregressive). In this manner, the multi-class output layer 148 enables the computing system 102 to predict a plurality of words or word pieces 152 for two or more of the plurality of patches 112 in parallel. This enables the transformer neural network 128 to generate the text representation of the text within the input image faster than another transformer neural network that employs an autoregressive decoder. The multi-class output layer 148 may also be implemented on one or more of a plurality of different hardware and platforms, including both central processing unit (CPU) and graphics processing unit (GPU) implementations. This flexibility can lead to further reduction in runtime latency relative to a conventional transformer neural network and enables the methods and systems disclosed herein to be implemented on platforms providing various services, including real-time cloud-based OCR.

[0035] The following paragraphs provide additional details regarding the training of the transformer neural network 128 of FIG. 1. In some examples, the transformer encoder 130 and the multi-class output layer 148 are trained together on a

text recognition task. In some such examples, and with reference now to FIG. 3, the computing system 102 is configured to, during a training phase, receive pre-training data 162. The pre-training data 162 includes, as input, a plurality of pre-training image frames 164 comprising pre-training text 166. For each pre-training image frame 164, the pre-training data 162 includes a ground-truth text representation 168 of the pre-training text 166. The transformer neural network 128 is trained on the pre-training data 162 to initialize the transformer encoder 130 and the multi-class output layer 148. Pre-training initializes the transformer neural network 128 to recognize patterns of pixels in the pre-training image frames 164 likely to correspond to a vocabulary word or word piece in the pre-training text 166.

[0036] As depicted in FIG. 3, the pre-training of the transformer neural network 128 is supervised by the provision of the ground truth text representation 168. The pre-training may be conducted using a suitable loss function, such as by cross-entropy loss. In some examples, a byte-pair encoding tokenizer is used to parse the ground-truth text representation 168 into words or word pieces. Front and back boundaries of a sentence are marked with a special token “[BOS]” and “[EOS]”, respectively. The tokens are converted into token IDs. In some examples, the output sequence is smaller than the transformer input 111. In some examples, fixed tokens are added to the end of the output sequence to pad the output to a fixed size of the encoder representation 146.

[0037] In some examples, the transformer neural network 128 is pre-trained on synthetic data 170. Advantageously, the computing system 102 and/or another computing system may be configured to rapidly generate a large volume of synthetic data 170. For example, a word processor may be automated to export images of ground-truth text in a variety of languages, styles, font sizes, and colors. This enables

the computing system 102 to initialize transformer neural network 128 on a large volume of pre-training data 162 (e.g., on the order of hundreds of millions of labeled textline images). The large volume of pre-training data enables the transformer neural network 128 to recognize more and broader instances of patterns. In another potential advantage of the present disclosure, and as described in more detail below, the pre-trained transformer neural network 128 may be fine-tuned for one or more tasks other than text recognition, such as text detection and text localization. The transformer neural network 128 may additionally or alternatively be fine-tuned to recognize printed and/or handwritten text, and additionally or alternatively recognize text in one or more of a plurality of languages.

[0038] FIG. 4 shows an example of fine-tuning the transformer neural network 128 for a text recognition task, such as optical character recognition. During the training phase (e.g., after the pre-training of FIG. 3), the computing system 102 is configured to receive fine-tuning data 172. The fine-tuning data 172 includes, as input, a plurality of fine-tuning image frames 174. The plurality of fine-tuning image frames 174 includes one or more of printed text 176 or handwritten text 178. For each fine-tuning image frame 174, the fine-tuning data 172 includes a ground-truth text representation 180 of the one or more of the printed text 176 or the handwritten text 178. In some examples, the ground-truth text representation 180 is human-generated. Accordingly, the fine-tuning data 172 may be selected from a smaller dataset than the pre-training data 162 of FIG. 3. The transformer neural network 128 is fine-tuned on the fine-tuning data 172 to generate a run-time text representation of run-time text in a run-time input image. In this manner, the transformer neural network 128 leverages parameters gleaned from the pre-training data to learn its runtime task from a relatively smaller fine-tuning dataset.

[0039] In some examples, after the text representation 104 is output at runtime, a user can provide feedback indicating that the text representation 104 was accurate or inaccurate. The feedback is then paired with the input image 108 as a feedback training data pair and used to conduct feedback training on the transformer neural network 128. As an illustrative example, a pharmacy may employ the computing system 102 to perform OCR on handwritten prescriptions (as input image frames). The pharmacy may provide feedback if there is an error in the text representation of the input image frame. The pharmacy may additionally or alternatively contact a doctor's office that issued the prescription at issue to learn the medication, dosage, etc., written on the prescription, which may become ground truth for the feedback training. In this manner, the feedback training may increase the accuracy of the transformer neural network.

[0040] In some examples, and with reference again to FIG. 2, the tokenized vocabulary 154 includes a plurality of vocabulary words or word pieces 152 in a plurality of languages, and the computing system 102 is configured to generate the text representation 104 in one or more of the plurality of languages. One transformer neural network 128 may be trained to recognize text in a plurality of languages using a tokenized vocabulary on the order of 200,000 word pieces (at least a portion of which may overlap from different languages), in one example. The transformer neural network 128 may additionally or alternatively recognize letter case for case-sensitive OCR and other case-sensitive applications using additional word pieces. For example, an English-language case-insensitive transformer neural network may have a vocabulary of approximately 30,000 word pieces, while a case-sensitive English-language transformer neural network may have a larger vocabulary of approximately

60,000 word pieces. This enables the transformer neural network to recognize letter case in input images.

[0041] FIG. 5 shows another example of fine-tuning the transformer neural network 128. As described in more detail below, the transformer neural network 128 may be fine-tuned for a text classification task. In the example of FIG. 5, the computing system 102 is configured to receive fine-tuning data 182 during the training phase (e.g., after the pre-training of FIG. 3). The fine-tuning data 182 includes, as input, a plurality of fine-tuning image frames. In some examples, the plurality of fine-tuning image frames are the same fine-tuning image frames 174 used to fine-tune the transformer neural network 128 for the text recognition task of FIG. 4. In other examples, the fine-tuning data 182 additionally or alternatively includes other image frames.

[0042] For each fine-tuning image frame, the fine-tuning data 182 includes a ground-truth text classification 184. The transformer neural network 128 is fine-tuned on the fine-tuning data 182 to output a run-time text classification for a run-time input image. In some examples, the run-time text classification and the ground-truth text classification 184 include one or more of text detection 186 or text localization 188. One example of a text detection task 186 includes identifying whether text is present in an input image. One example of a text localization task 188 includes identifying a location of text within the input image. In this manner, the initialized transformer neural network 128 may be leveraged for one or more of a variety of text-related tasks.

[0043] With reference now to FIG. 6, a flowchart is illustrated depicting an example method 600 for generating a text representation of text contained within an input image. The following description of method 600 is provided with reference to the software and hardware components described above and shown in FIGS. 1-5 and

7, and the method steps in method 600 will be described with reference to corresponding portions of FIGS. 1-5 and 7 below. It will be appreciated that method 600 also may be performed in other contexts using other suitable hardware and software components.

[0044] It will be appreciated that the following description of method 600 is provided by way of example and is not meant to be limiting. It will be understood that various steps of method 600 can be omitted or performed in a different order than described, and that the method 600 can include additional and/or alternative steps relative to those illustrated in FIG. 6 without departing from the scope of this disclosure.

[0045] The method 600 includes, at 602, obtaining an input image including text. For example, the computing system 102 of FIG. 1 is configured to obtain the input image 108 including text 106, such as by receiving the input image 108 from the client 110. At 604, the method 600 includes decomposing the input image into a plurality of patches. For example, the input image 108 of FIG. 1 is decomposed into a plurality of patches 112. The plurality of patches are arranged in an input sequence at 606. At 608, the method 600 includes generating a patch embedding for each of the plurality of patches via a patch embedding layer. The patch embedding includes a vector representation of the patch. For example, the computing system 102 is configured to generate patch embedding 114 for each of the plurality of patches 112. This converts the input image into a format that can be processed by the transformer neural network 128.

[0046] At 610, the method 600 includes generating a position embedding for each of the plurality of patches via a position embedding layer. The position embedding includes a position of the patch in the input sequence. For example, the

computing system 102 is configured to generate position embedding 124 for each of the plurality of patches 112. The position embedding 124 provides the transformer neural network 128 with a learnable representation of the position of each patch 112 in the input sequence.

[0047] The method 600 further includes, at 612, providing the position embeddings and patch embeddings for each of the plurality of input patches according to the input sequence to a transformer encoder to thereby cause the transformer encoder to output an encoder representation of the input sequence. For example, the computing system 102 is configured to generate the encoder representation 146 of FIG. 2 using attention module 132 to selectively concentrate weight on particular values within the transformer input 111 and using the one or more feed-forward layers 136 to fit the encoder representation 146 for the multi-class output layer 148.

[0048] At 614, the method 600 includes providing the encoder representation to a multi-class output layer configured with a plurality of nodes, each node representing a word in a tokenized vocabulary, to thereby cause the multi-class output layer to output a predicted vocabulary word for the encoder representation of the input sequence. For example, the computing system 102 is configured to provide the encoder representation 146 to the multi-class output layer 148. The multi-class output layer generates a probability distribution over the tokenized vocabulary 154, which is used to generate the text representation 104.

[0049] The method 600 further includes, at 616, generating a text representation of the text of the input image including the predicted vocabulary word. For example, the computing system 102 of FIGS. 1-2 may be configured to generate the text representation 104 by taking an index of a highest probability score output by the multi-class output layer 148 for each dimension of the encoder representation 146.

[0050] The above-described systems and methods may be used to generate a text representation of text contained within an input image. A multi-class output layer is introduced in a transformer-based neural network downstream of a transformer encoder, in place of an autoregressive transformer decoder. The multi-class output layer enables the transformer neural network to predict a plurality of words or word pieces in parallel. This reduces an amount of time to generate the text representation relative to the use of the autoregressive transformer decoder, while maintaining similar accuracy. In one example experiment, the transformer neural network described herein recognized 832 tokens/second while a conventional transformer recognized 89 tokens/second. The transformer neural network may be pre-trained to leverage a large volume of data (e.g., synthetic data). This enables the transformer neural network to broadly recognize text patterns in an input image. The pre-trained transformer neural network may then be fine-tuned for one or more of a plurality of tasks, including text recognition (which may be case-sensitive and/or multilingual), text detection, and text localization.

[0051] In some embodiments, the methods and processes described herein may be tied to a computing system of one or more computing devices. In particular, such methods and processes may be implemented as a computer-application program or service, an application-programming interface (API), a library, and/or other computer-program product.

[0052] FIG. 7 schematically shows an example of a computing system 700 that can enact one or more of the devices and methods described above. Computing system 700 is shown in simplified form. Computing system 700 may take the form of one or more personal computers, server computers, tablet computers, home-entertainment computers, network computing devices, gaming devices, mobile

computing devices, mobile communication devices (e.g., smart phone), wearable computing devices such as smart wristwatches and head mounted augmented reality devices, and/or other computing devices. In some examples, the computing system 700 may embody the computing system 102 and/or the client 110 of FIG. 1.

[0053] The computing system 700 includes a logic processor 702, volatile memory 704, and a non-volatile storage device 706. The computing system 700 may optionally include a display subsystem 708, input subsystem 710, communication subsystem 712, and/or other components not shown in FIG. 7.

[0054] Logic processor 702 includes one or more physical devices configured to execute instructions. For example, the logic processor may be configured to execute instructions that are part of one or more applications, programs, routines, libraries, objects, components, data structures, or other logical constructs. Such instructions may be implemented to perform a task, implement a data type, transform the state of one or more components, achieve a technical effect, or otherwise arrive at a desired result.

[0055] The logic processor may include one or more physical processors (hardware) configured to execute software instructions. Additionally or alternatively, the logic processor may include one or more hardware logic circuits or firmware devices configured to execute hardware-implemented logic or firmware instructions. Processors of the logic processor 702 may be single-core or multi-core, and the instructions executed thereon may be configured for sequential, parallel, and/or distributed processing. Individual components of the logic processor optionally may be distributed among two or more separate devices, which may be remotely located and/or configured for coordinated processing. Aspects of the logic processor may be virtualized and executed by remotely accessible, networked computing devices

configured in a cloud-computing configuration. In such a case, these virtualized aspects are run on different physical logic processors of various different machines, it will be understood.

[0056] Non-volatile storage device 706 includes one or more physical devices configured to hold instructions executable by the logic processors to implement the methods and processes described herein. When such methods and processes are implemented, the state of non-volatile storage device 706 may be transformed—e.g., to hold different data.

[0057] Non-volatile storage device 706 may include physical devices that are removable and/or built in. Non-volatile storage device 706 may include optical memory (e.g., CD, DVD, HD-DVD, Blu-Ray Disc, etc.), semiconductor memory (e.g., ROM, EPROM, EEPROM, FLASH memory, etc.), and/or magnetic memory (e.g., hard-disk drive, floppy-disk drive, tape drive, MRAM, etc.), or other mass storage device technology. Non-volatile storage device 706 may include nonvolatile, dynamic, static, read/write, read-only, sequential-access, location-addressable, file-addressable, and/or content-addressable devices. It will be appreciated that non-volatile storage device 706 is configured to hold instructions even when power is cut to the non-volatile storage device 706.

[0058] Volatile memory 704 may include physical devices that include random access memory. Volatile memory 704 is typically utilized by logic processor 702 to temporarily store information during processing of software instructions. It will be appreciated that volatile memory 704 typically does not continue to store instructions when power is cut to the volatile memory 704.

[0059] Aspects of logic processor 702, volatile memory 704, and non-volatile storage device 706 may be integrated together into one or more hardware-logic

components. Such hardware-logic components may include field-programmable gate arrays (FPGAs), program- and application-specific integrated circuits (ASIC / ASICs), program- and application-specific standard products (PSSP / ASSPs), system-on-a-chip (SOC), and complex programmable logic devices (CPLDs), for example.

[0060] The terms “module” and “program” may be used to describe an aspect of computing system 700 typically implemented in software by a processor to perform a particular function using portions of volatile memory, which function involves transformative processing that specially configures the processor to perform the function. Thus, a module or program may be instantiated via logic processor 702 executing instructions held by non-volatile storage device 706, using portions of volatile memory 704. It will be understood that different modules and/or programs may be instantiated from the same application, service, code block, object, library, routine, API, function, etc. Likewise, the same module and/or program may be instantiated by different applications, services, code blocks, objects, routines, APIs, functions, etc. The terms “module” and “program” may encompass individual or groups of executable files, data files, libraries, drivers, scripts, database records, etc.

[0061] It will be appreciated that a “service”, as used herein, is an application program executable across multiple user sessions. A service may be available to one or more system components, programs, and/or other services. In some implementations, a service may run on one or more server-computing devices.

[0062] When included, display subsystem 708 may be used to present a visual representation of data held by non-volatile storage device 706. The visual representation may take the form of a GUI. As the herein described methods and processes change the data held by the non-volatile storage device, and thus transform

the state of the non-volatile storage device, the state of display subsystem 708 may likewise be transformed to visually represent changes in the underlying data. Display subsystem 708 may include one or more display devices utilizing virtually any type of technology. Such display devices may be combined with logic processor 702, volatile memory 704, and/or non-volatile storage device 706 in a shared enclosure, or such display devices may be peripheral display devices.

[0063] When included, input subsystem 710 may comprise or interface with one or more user-input devices such as a keyboard, mouse, touch screen, or game controller. In some examples, the input subsystem may comprise or interface with selected natural user input (NUI) componentry. Such componentry may be integrated or peripheral, and the transduction and/or processing of input actions may be handled on- or off-board. Example NUI componentry may include a microphone for speech and/or voice recognition; an infrared, color, stereoscopic, and/or depth camera for machine vision and/or gesture recognition; a head tracker, eye tracker, accelerometer, and/or gyroscope for motion detection and/or intent recognition; as well as electric-field sensing componentry for assessing brain activity; and/or any other suitable sensor.

[0064] When included, communication subsystem 712 may be configured to communicatively couple various computing devices described herein with each other, and with other devices. Communication subsystem 712 may include wired and/or wireless communication devices compatible with one or more different communication protocols. For example, the communication subsystem may be configured for communication via a wireless telephone network, or a wired or wireless local- or wide-area network. In some examples, the communication

subsystem may allow computing system 700 to send and/or receive messages to and/or from other devices via a network such as the Internet.

[0065] The following paragraphs provide additional support for the claims of the subject application. One aspect provides a computing system, comprising: a processor; and a memory storing instructions executable by the processor to obtain an input image including text for optical character recognition; decompose the input image into a plurality of patches; arrange the plurality of patches in an input sequence; generate a patch embedding for each of the plurality of patches via a patch embedding layer, the patch embedding including a vector representation of the patch; generate a position embedding for each of the plurality of patches via a position embedding layer, the position embedding indicating a position of the patch in the input sequence; provide the position embeddings and patch embeddings for each of the plurality of input patches according to the input sequence to a transformer encoder of a neural network to thereby cause the transformer encoder to output an encoder representation of the input sequence; provide the encoder representation to a multi-class output layer of the neural network configured with a plurality of nodes, each node representing a word or a word piece in a tokenized vocabulary, to thereby cause the multi-class output layer to output a predicted vocabulary word or word piece for the encoder representation of the input sequence; and generate a text representation of the text of the input image including the predicted vocabulary word or word piece. In some examples, the instructions are additionally or alternatively executable to, during a training phase: receive pre-training data including, as input, a plurality of pre-training image frames including pre-training text, and for each pre-training image frame, a text representation of the pre-training text; and train the neural network on the pre-training data to initialize the transformer encoder and the multi-class output layer. In some

examples, the instructions are additionally or alternatively executable to, during the training phase: receive fine-tuning data including, as input, a plurality of fine-tuning image frames including one or more of printed text or handwritten text, and for each fine-tuning image frame, a ground-truth text representation of the one or more of the printed text or the handwritten text; and fine-tune the neural network on the fine-tuning data to generate a run-time text representation of run-time text in a run-time input image. In some examples, the instructions are additionally or alternatively executable to, during the training phase: receive fine-tuning data including, as input a plurality of fine-tuning image frames including one or more of printed text or handwritten text, and for each fine-tuning image frame, a ground-truth text classification; and fine-tune the neural network on the fine-tuning data to output a run-time text classification for a run-time input image. In some examples, the ground-truth text classification additionally or alternatively includes one or more of text detection or text localization. In some examples, the transformer encoder additionally or alternatively includes a multi-head attention function and one or more feed-forward layers. In some examples, the transformer encoder additionally or alternatively includes a residual connection layer and a normalization layer between the multi-head attention function and the one or more feed-forward layers, and an additional residual connection layer and an additional normalization layer downstream of the one or more feed-forward layers. In some examples, the transformer encoder and the multi-class output layer are additionally or alternatively included in a transformer neural network. In some examples, the tokenized vocabulary additionally or alternatively includes a plurality of vocabulary words or word pieces in a plurality of languages, and the instructions executable to generate the text representation are additionally or alternatively executable to generate the text representation in one or more of the

plurality of languages. In some examples, the instructions executable to obtain the input image additionally or alternatively include instructions executable to receive the input image from a client, and the instructions are additionally or alternatively executable to output the text representation to the client. In some examples, the instructions are additionally or alternatively executable to normalize the input image to a normalized image resolution; and decompose the normalized input image into the plurality of patches. In some examples, the multi-class output layer additionally or alternatively includes a softmax function. In some examples, the instructions are additionally or alternatively executable to output the predicted vocabulary word or word piece for two or more of the plurality of patches in parallel. In some examples, the instructions are additionally or alternatively executable to provide an image classification token and a distillation token to the transformer encoder.

[0066] Another aspect provides, at a computing system, a method for generating a text representation of text contained within an input image, the method comprising: obtaining an input image including text; decomposing the input image into a plurality of patches; arranging the plurality of patches in an input sequence; generating a patch embedding for each of the plurality of patches via a patch embedding layer, the patch embedding including a vector representation of the patch; generating a position embedding for each of the plurality of patches via a position embedding layer, the position embedding indicating a position of the patch in the input sequence; providing the position embeddings and patch embeddings for each of the plurality of input patches according to the input sequence to a transformer encoder to thereby cause the transformer encoder to output an encoder representation of the input sequence; providing the encoder representation to a multi-class output layer configured with a plurality of nodes, each node representing a word or a word piece in

a tokenized vocabulary, to thereby cause the multi-class output layer to output a predicted vocabulary word or word piece for the encoder representation of the input sequence; and generating a text representation of the text of the input image including the predicted vocabulary word or word piece. In some examples, the method additionally or alternatively includes, during a training phase: receiving pre-training data including, as input, a plurality of pre-training image frames including pre-training text, and for each pre-training image frame, a text representation of the pre-training text; and training the neural network on the pre-training data to initialize the transformer encoder and the multi-class output layer. In some examples, the method additionally or alternatively includes, during the training phase: receiving fine-tuning data including, as input, a plurality of fine-tuning image frames including one or more of printed text or handwritten text, and for each fine-tuning image frame, a ground-truth text representation of the one or more of the printed text or the handwritten text; and fine-tuning the neural network on the fine-tuning data to generate a run-time text representation of run-time text in a run-time input image. In some examples, the method additionally or alternatively includes, during the training phase: receiving fine-tuning data including, as input a plurality of fine-tuning image frames including one or more of printed text or handwritten text, and for each fine-tuning image frame, a ground-truth text classification; and fine-tuning the neural network on the fine-tuning data to output a run-time text classification for a run-time input image. In some examples, obtaining the input image additionally or alternatively includes receiving the input image from a client, the method further comprising outputting the text representation to the client.

[0067] Another aspect provides a computing system, comprising: a processor; and a memory storing instructions executable by the processor to receive, from a

client, an input image including text; decompose the input image into a plurality of patches; arrange the plurality of patches in an input sequence; generate a patch embedding for each of the plurality of patches via a patch embedding layer, the patch embedding including a vector representation of the patch; generate a position embedding for each of the plurality of patches via a position embedding layer, the position embedding indicating a position of the patch in the input sequence; provide the position embeddings and patch embeddings for each of the plurality of input patches according to the input sequence to a transformer encoder of a neural network to thereby cause the transformer encoder to output an encoder representation of the input sequence; provide the encoder representation to a multi-class output layer of the neural network configured with a plurality of nodes, each node representing a word or a word piece in a tokenized vocabulary, to thereby cause the multi-class output layer to output a predicted vocabulary word or word piece for the encoder representation of the input sequence; generate a text representation of the text of the input image including the predicted vocabulary word or word piece; and output the text representation to the client.

[0068] It will be understood that the configurations and/or approaches described herein are exemplary in nature, and that these specific embodiments or examples are not to be considered in a limiting sense, because numerous variations are possible. The specific routines or methods described herein may represent one or more of any number of strategies. As such, various acts illustrated and/or described may be performed in the sequence illustrated and/or described, in other sequences, in parallel, or omitted. Likewise, the order of the above-described processes may be changed.

[0069] The subject matter of the present disclosure includes all novel and non-obvious combinations and sub-combinations of the various processes, systems and configurations, and other features, functions, acts, and/or properties disclosed herein, as well as any and all equivalents thereof.

[0070] Further, it will be appreciated that the terms “includes,” “including,” “has,” “contains,” variants thereof, and other similar words used in either the detailed description or the claims are intended to be inclusive in a manner similar to the term “comprising” as an open transition word without precluding any additional or other elements.

CLAIMS:

1. A computing system, comprising:
 - a processor; and
 - a memory storing instructions executable by the processor to
 - obtain an input image including text for optical character recognition;
 - decompose the input image into a plurality of patches;
 - arrange the plurality of patches in an input sequence;
 - generate a patch embedding for each of the plurality of patches via a patch embedding layer, the patch embedding including a vector representation of the patch;
 - generate a position embedding for each of the plurality of patches via a position embedding layer, the position embedding indicating a position of the patch in the input sequence;
 - provide the position embeddings and patch embeddings for each of the plurality of input patches according to the input sequence to a transformer encoder of a neural network to thereby cause the transformer encoder to output an encoder representation of the input sequence;
 - provide the encoder representation to a multi-class output layer of the neural network configured with a plurality of nodes, each node representing a word or a word piece in a tokenized vocabulary, to thereby cause the multi-class output layer to output a predicted vocabulary word or word piece for the encoder representation of the input sequence; and
 - generate a text representation of the text of the input image including the predicted vocabulary word or word piece.

2. The computing system of claim 1, wherein the instructions are further executable to, during a training phase:

receive pre-training data including,

as input, a plurality of pre-training image frames including pre-training text, and

for each pre-training image frame, a text representation of the pre-training text; and

train the neural network on the pre-training data to initialize the transformer encoder and the multi-class output layer.

3. The computing system of claim 2, wherein the instructions are further executable to, during the training phase:

receive fine-tuning data including,

as input, a plurality of fine-tuning image frames including one or more of printed text or handwritten text, and

for each fine-tuning image frame, a ground-truth text representation of the one or more of the printed text or the handwritten text; and

fine-tune the neural network on the fine-tuning data to generate a run-time text representation of run-time text in a run-time input image.

4. The computing system of claim 2, wherein the instructions are further executable to, during the training phase:

receive fine-tuning data including,

as input a plurality of fine-tuning image frames including one or more of printed text or handwritten text, and

for each fine-tuning image frame, a ground-truth text classification;
and

fine-tune the neural network on the fine-tuning data to output a run-time text classification for a run-time input image.

5. The computing system of claim 4, wherein the ground-truth text classification includes one or more of text detection or text localization.

6. The computing system of claim 1, wherein the transformer encoder includes a multi-head attention function and one or more feed-forward layers.

7. The computing system of claim 6, wherein the transformer encoder further includes a residual connection layer and a normalization layer between the multi-head attention function and the one or more feed-forward layers, and an additional residual connection layer and an additional normalization layer downstream of the one or more feed-forward layers.

8. The computing system of claim 1, wherein the transformer encoder and the multi-class output layer are included in a transformer neural network.

9. The computing system of claim 1, wherein the tokenized vocabulary includes a plurality of vocabulary words or word pieces in a plurality of languages, and wherein the instructions executable to generate the text representation are further executable to generate the text representation in one or more of the plurality of languages.

10. The computing system of claim 1, wherein the instructions executable to obtain the input image include instructions executable to receive the input image from a client, and wherein the instructions are further executable to output the text representation to the client.

11. The computing system of claim 1, wherein the instructions are further executable to:

normalize the input image to a normalized image resolution; and

decompose the normalized input image into the plurality of patches.

12. The computing system of claim 1, wherein the multi-class output layer includes a softmax function.

13. The computing system of claim 1, wherein the instructions are further executable to output the predicted vocabulary word or word piece for two or more of the plurality of patches in parallel.

14. The computing system of claim 1, wherein the instructions are further executable to provide an image classification token and a distillation token to the transformer encoder.

15. At a computing system, a method for generating a text representation of text contained within an input image, the method comprising:

obtaining an input image including text;

decomposing the input image into a plurality of patches;

arranging the plurality of patches in an input sequence;

generating a patch embedding for each of the plurality of patches via a patch embedding layer, the patch embedding including a vector representation of the patch;

generating a position embedding for each of the plurality of patches via a position embedding layer, the position embedding indicating a position of the patch in the input sequence;

providing the position embeddings and patch embeddings for each of the plurality of input patches according to the input sequence to a transformer encoder to thereby cause the transformer encoder to output an encoder representation of the input sequence;

providing the encoder representation to a multi-class output layer configured with a plurality of nodes, each node representing a word or a word piece in a tokenized vocabulary, to thereby cause the multi-class output layer to output a predicted vocabulary word or word piece for the encoder representation of the input sequence; and

generating a text representation of the text of the input image including the predicted vocabulary word or word piece.

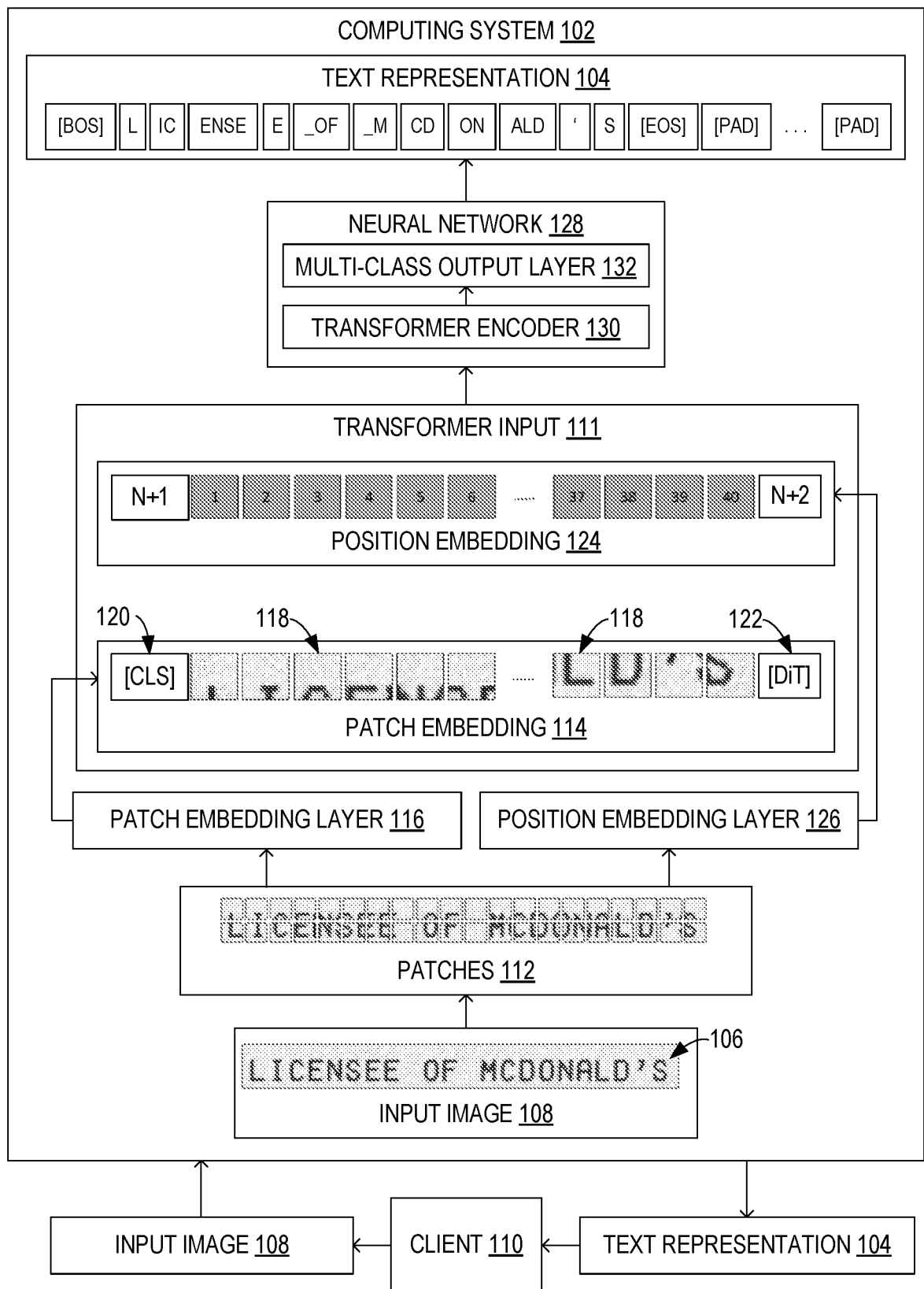


FIG. 1

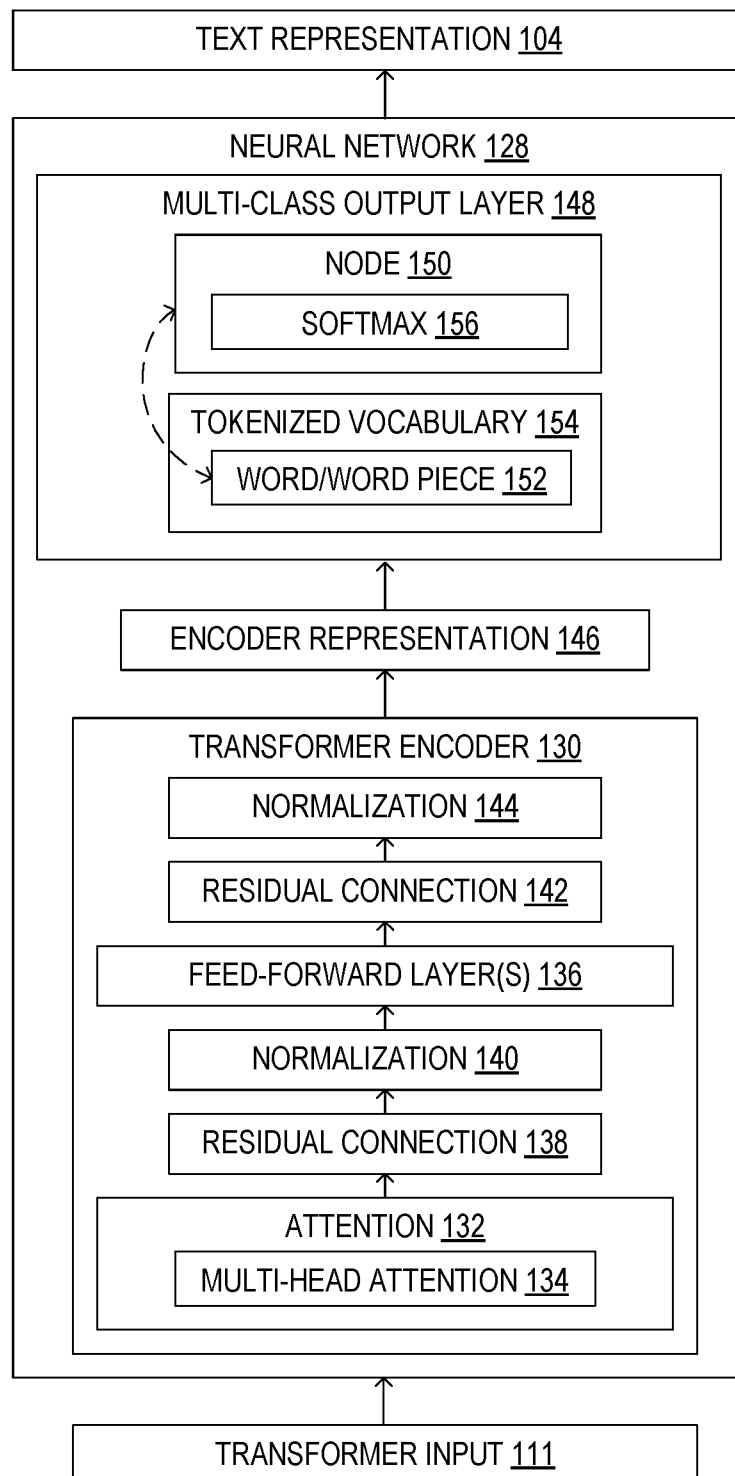


FIG. 2

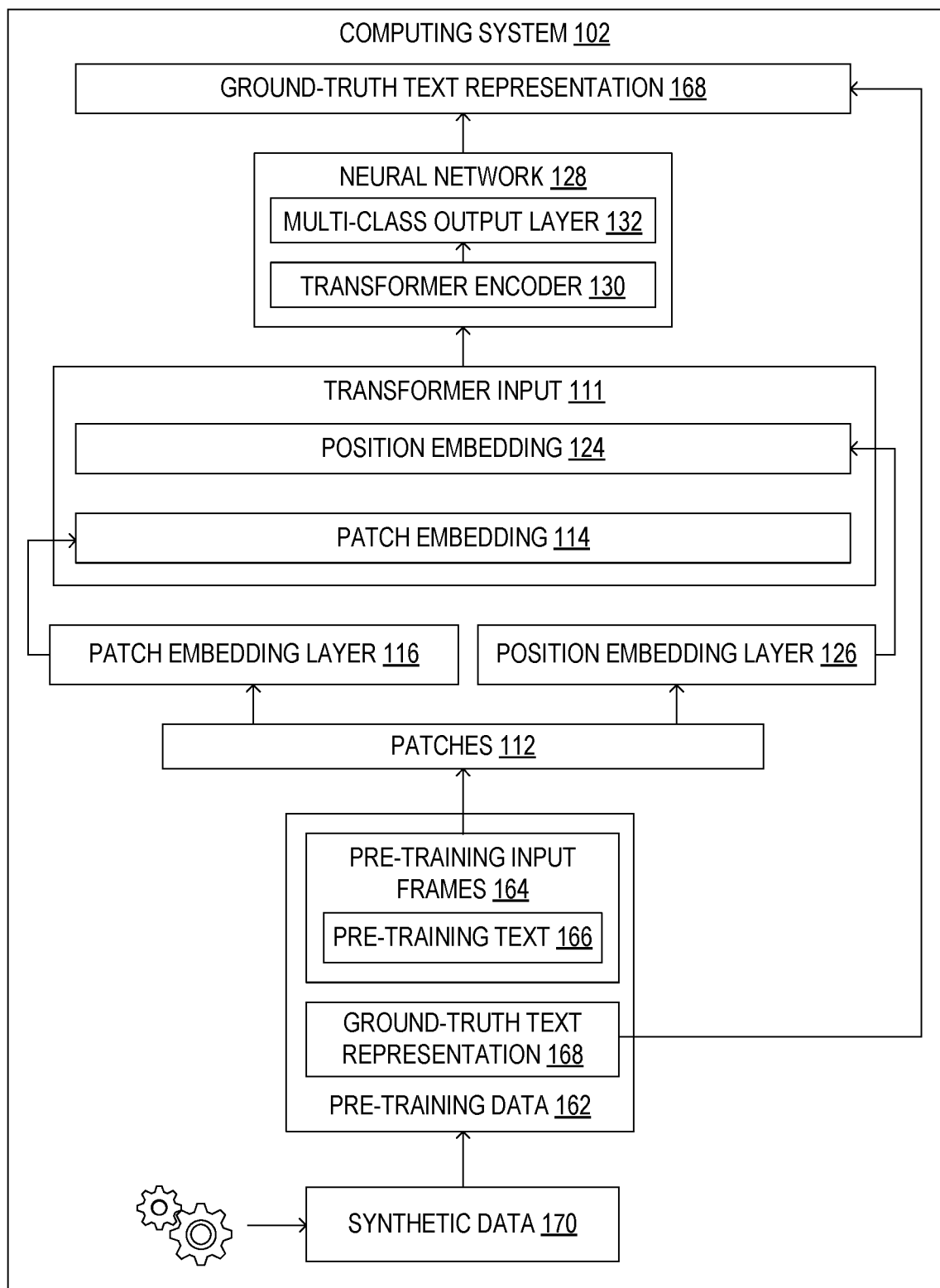


FIG. 3

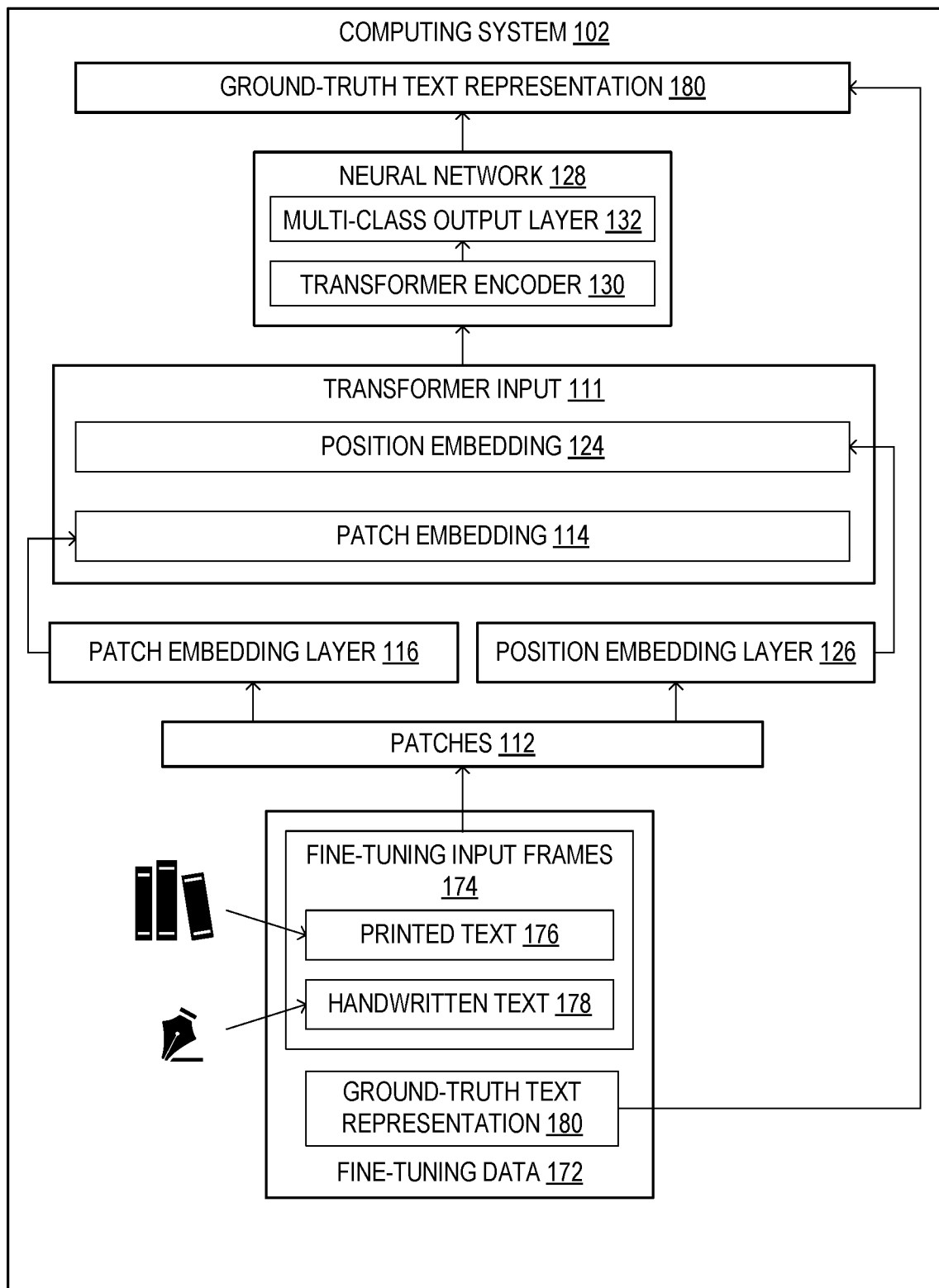


FIG. 4

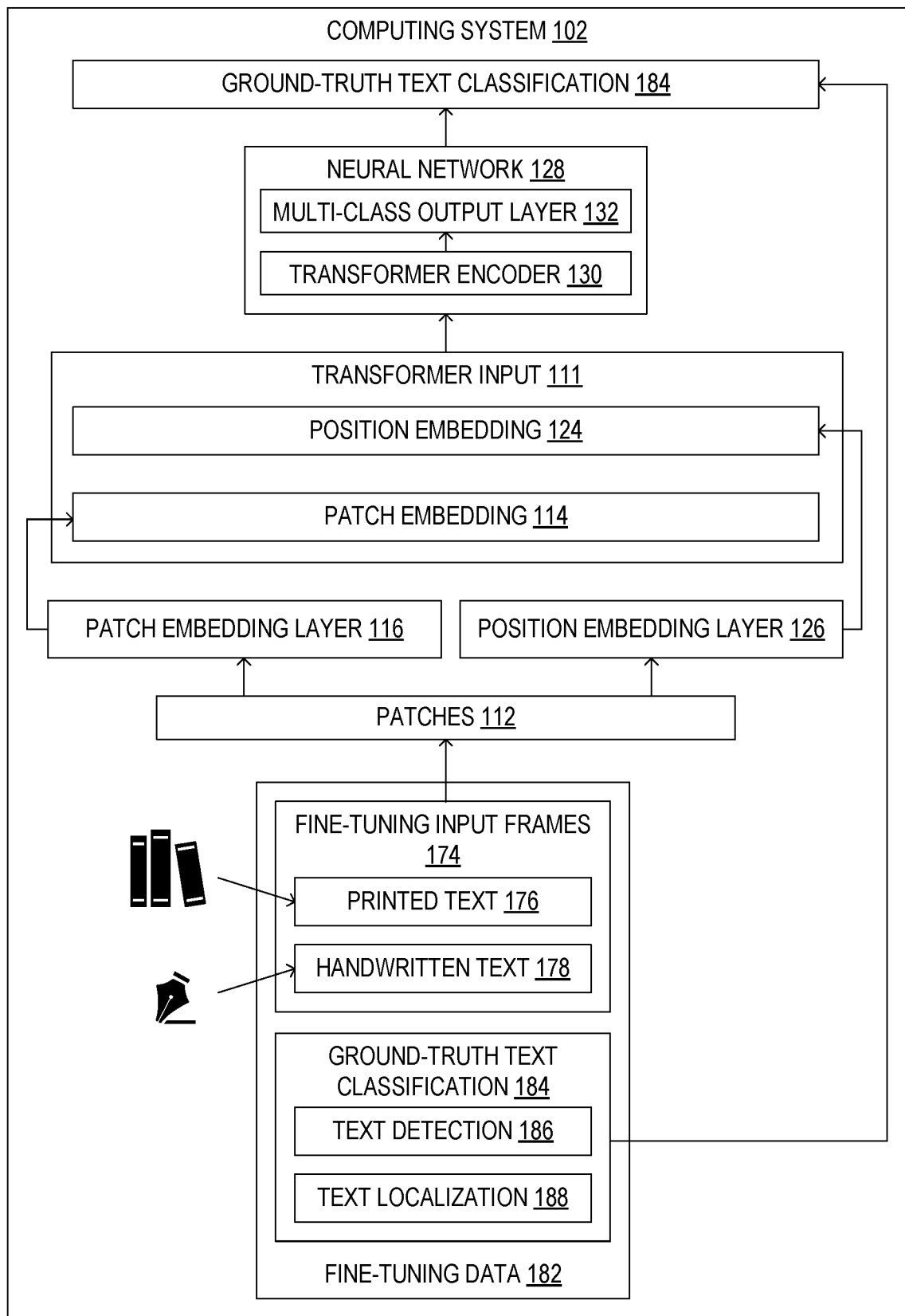


FIG. 5

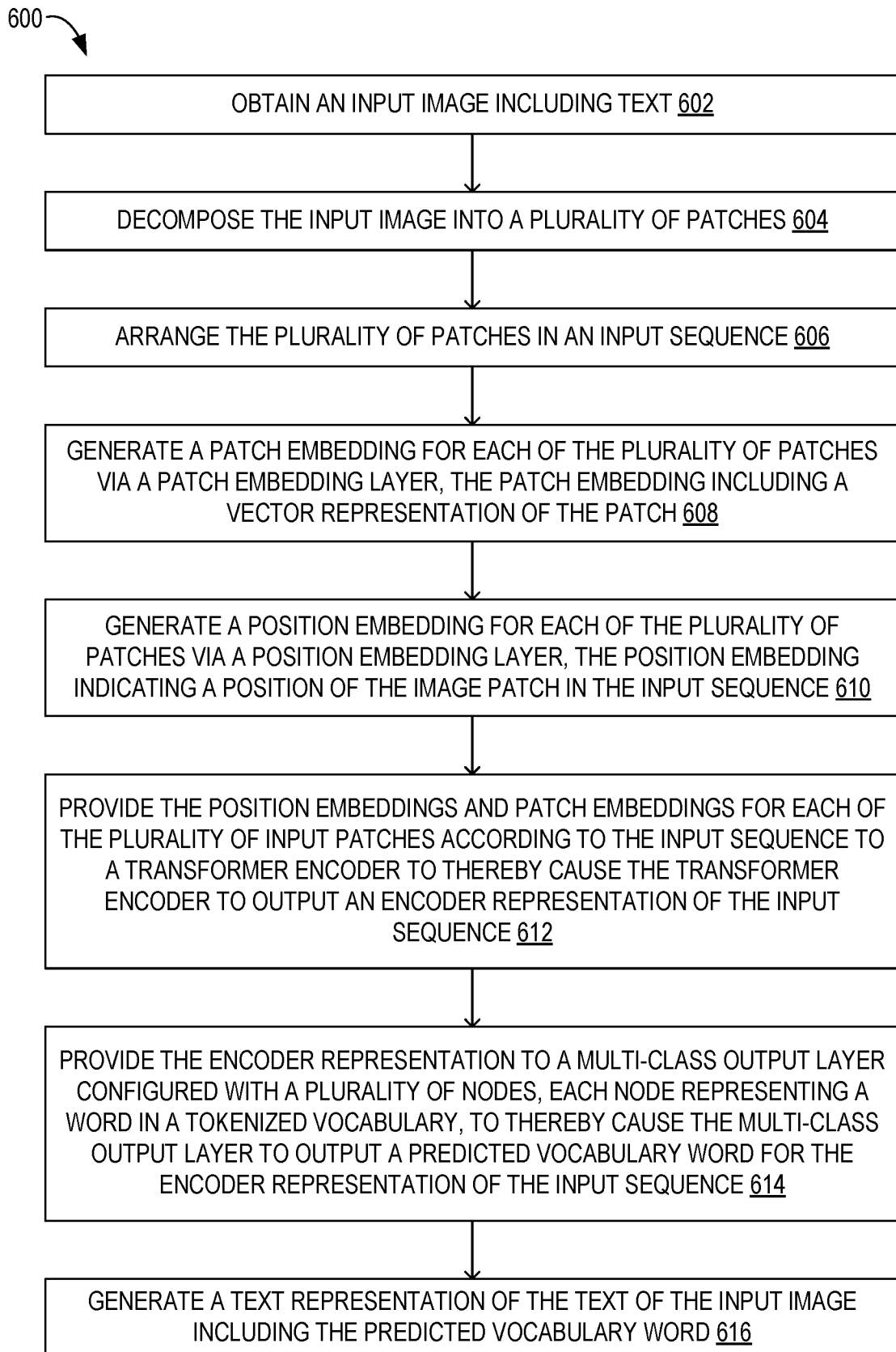


FIG. 6

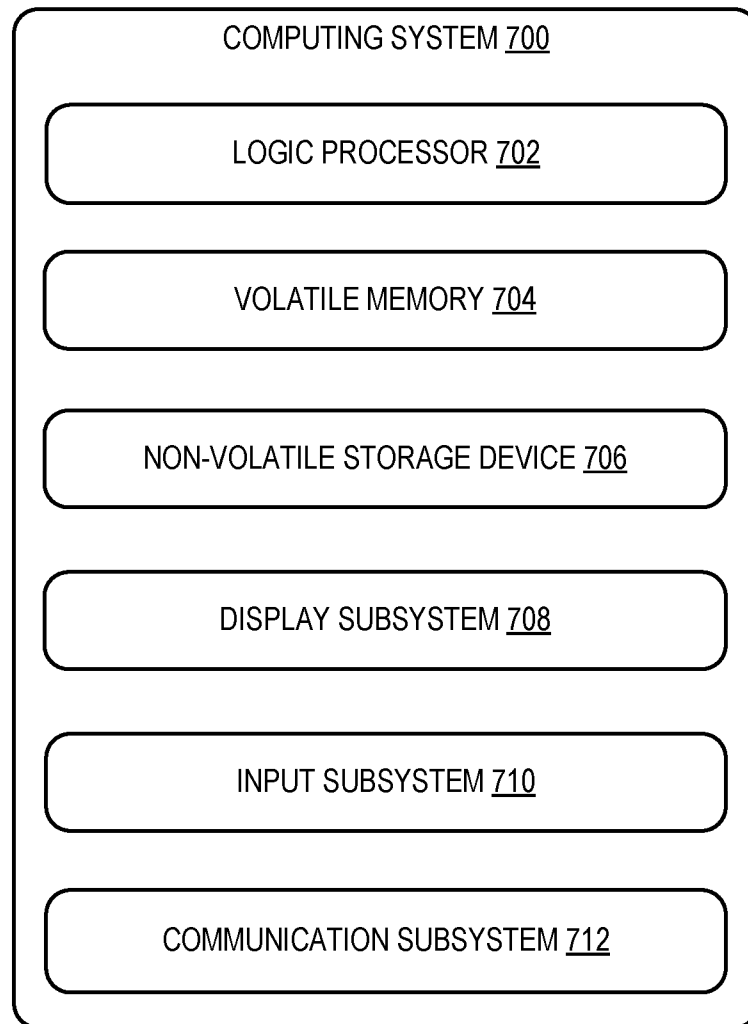


FIG. 7

INTERNATIONAL SEARCH REPORT

International application No

PCT/CN2022/090014

A. CLASSIFICATION OF SUBJECT MATTER

INV. G06V10/82 G06V30/18

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)

G06V

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, WPI Data**C. DOCUMENTS CONSIDERED TO BE RELEVANT**

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	MINGHAO LI ET AL: "TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models", ARXIV.ORG, CORNELL UNIVERSITY LIBRARY, 201 OLIN LIBRARY CORNELL UNIVERSITY ITHACA, NY 14853, 21 September 2021 (2021-09-21), XP091056588, Sections 1-3 ----- -/--	1-15



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

18 October 2022

Date of mailing of the international search report

28/10/2022

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

Borges, Pedro

INTERNATIONAL SEARCH REPORT

International application No

PCT/CN2022/090014

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	ASHISH VASWANI ET AL: "Attention Is All You Need", ARXIV.ORG, CORNELL UNIVERSITY LIBRARY, 201 OLIN LIBRARY CORNELL UNIVERSITY ITHACA, NY 14853 , 6 December 2017 (2017-12-06), XP080973732, Retrieved from the Internet: URL:https://arxiv.org/abs/1706.03762v5 [retrieved on 2017-12-06] the whole document -----	1-15
A	HUGO TOUVRON ET AL: "Training data-efficient image transformers & distillation through attention", ARXIV.ORG, CORNELL UNIVERSITY LIBRARY, 201 OLIN LIBRARY CORNELL UNIVERSITY ITHACA, NY 14853, 15 January 2021 (2021-01-15), XP081859745, the whole document -----	1-15