



Multi-Modal Image and Text Processing Using a Transformer for Sub-Figure Sequence Classification with or without a convolutional pre-processor.

Bachelor's Project Thesis

Filipe Laitenberger, s3894479, f.m.de.sousa.horta.osorio.laitenberger@student.rug.nl,
Supervisor: Prof. Dr. Lambert Schomaker

Abstract: Recent findings in deep learning research indicate that the successful Transformer approach for text processing can also be used in the image domain, using Vision Transformers (ViTs). Compared to the common convolutional paradigm, the images are tokenized in a surprisingly primitive patch-wise fashion. This study introduces a multi-modal transformer architecture for image and text processing with a single encoder. Two variations of a transformer-based model were compared: a standard one, taking the raw, complete input image as input, and a variant with a convolutional pre-processor. Both variants were tested on a task with multiple objects combined in one image in a 2x2 grid alongside a text sequence. The model should decide whether the text sequence describes the correct 'reading order' of the objects shown in the image. Results showed that a transformer with a convolution-based pre-processing layer performed significantly better (about 96-98% accuracy) than the plain Transformer-based model (about 92% accuracy). Apparently, the Transformer is supported by the learned feature maps, increasing the reliability of the correlations needed for the sequentialization task. In turn, the results support the hypothesis that ViTs are inferior to CNN architectures regarding training time and data set size required for successful generalization. ViTs would need to overcome their inductive bias related to patch extraction, which, in its pure form, lacks appropriate feature extraction within patches. Further research could also examine whether the proposed hybrid architecture has advantages in more sophisticated data sets such as Visual Genome and tasks such as Visual Question Answering.

1 Introduction

In deep learning, processing text has evolved substantially in recent years. Contemporary research used to focus mainly on Recurrent Neural Networks (RNNs) [Rumelhart et al., 1985, Jordan, 1986] which can process input sequences of varying lengths. Refined such models that aim to master some of the shortcomings of traditional RNNs (e.g., the diminishing gradient problem) used to be state-of-the-art in text processing. The most prominent of these is the Long-Short-Term-Memory Network (LSTM) [Hochreiter and Schmidhuber, 1997]. However, more recent research in the field has shifted this focus to Attention mechanisms [Bahdanau et al., 2014], which led to the introduction of Transformer models [Vaswani et al., 2017]. The latter do not suffer from the shortcomings

predominant in RNNs, are computationally much more efficient to train, and perform substantially better on several tasks, including translation and classification.

Likewise, image processing has adopted the Convolutional Neural Network (CNN) [LeCun et al., 1999] as its state-of-the-art, which uses convolutional filters to detect features such as edges. These network architectures significantly outperform and require substantially fewer computational resources to train than ordinary Multi-Layer Perceptrons (MLPs) [Rosenblatt, 1960, Werbos, 1994]. Recently, research has been conducted on pure attention-based models that process images, namely the Vision Transformer (ViT), reaching performances similar to CNNs in

classification tasks on specific data sets. Instead of computing attention between every pair of pixels, the input image is split into non-overlapping patches that are flattened and linearly projected [Dosovitskiy et al., 2020].

However, one could hypothesize that there exist some apparent downsides to the ViT approach. Particularly, patch extraction complemented with positional encoding resembles a built-in inductive bias. By design, the model resorts to a specific way of looking at its input, splitting it into fixed-size regions, linearizing those, and giving them an order [Dosovitskiy et al., 2020]. This mechanism corresponds to how humans perceive by scanning a scene from one side to the other [Salvucci and Anderson, 1998, Noton and Stark, 1971, Reisberg, 2015]. However, the model might be limited at this point as it cannot find an optimal way of processing the input in a self-organizing way. Hence, a significant part of the model depends on the design choices of its creators. Another crucial drawback might be that attention can only be computed between patches, not within them. Thus, for computing features within patches, the model solely relies on a linear projection, which seems to be far less than optimal. These disadvantages seem to be reflected in ViT’s inability to generalize well on small to medium-sized data sets. Instead, the researchers who proposed the architecture trained it on over 300 mil. images before it reached performances comparable to CNNs [Dosovitskiy et al., 2020].

Apart from that, research has been conducted on multi-modality, i.e., simultaneously processing different (e.g., textual and visual) information formats. Present multi-modal neural networks share a common characteristic: they are comprised of multiple different modules for different modalities, e.g., a transformer/LSTM module for text processing and a CNN/transformer module for image processing [Hu and Singh, 2021, Tsai et al., 2019, Yu et al., 2019, Prakash et al., 2021, Huang et al., 2020].

A research gap becomes apparent here. With the intent of drawing connections between text tokens and image parts, a multi-modal architecture might benefit from the ability to perform self-attention on the entire input using a shared Transformer module. Apart from learning correlations within the respective modality, it could learn what image

parts correlate with which textual tokens and vice-versa. Aside from the downsides of image patchization, feeding two modalities into one Transformer would seem powerful here.

Henceforth, for the present research, a unified Transformer model shall be proposed that takes the current state-of-the-art further. Hence, it uses one module to process two modalities, i.e., textual and visual input.

On the other hand, it would seem that mere patchization and linear projection of the untouched input image is not enough to extract features properly. Instead, the latter would have to be learned in a self-organizing way, before the image is split into patches. A convolutional pre-processor would be suitable here. Therefore, the proposed model shall be tested in two conditions, producing two variants of the architecture in question. Specifically, as described above, a version that solely relies on patchization and attention for computing image features shall be contrasted with a variant that contains the aforementioned convolutional pre-processor. Through this, it shall be evaluated whether convolution extracts features more reliably than only linearly projecting patches and performing self-attention on them. A priori, it would seem that creating feature maps using convolutional filters would enhance the input fed into the Transformer. In comparison, mere patchization of the image would risk an inability to generalize as patches are only similarly embedded without computing features or attention within them. Comparison experiments will be conducted on the MNIST [LeCun et al.] and Fashion MNIST [Xiao et al., 2017] data sets featuring handwritten digits and clothing items respectively. However, a four-quadrant sub-figure classification task was designed to make it more challenging and give both approaches a chance to show their ability to handle sequential image analysis. In both conditions, a hyperparameter optimization search shall be carried out so that the amount of effort is comparable between the methods, making up for a fair evaluation.

2 Related Background

For brevity, basic Neural Networks, specifically Multilayer Perceptrons and Convolutional Neural

Networks, shall not be covered at this point. The reader is directed toward appendix A and B respectively for further details on these two topics.

Former research in text processing focused mainly on recurrent neural networks (RNNs) [Rumelhart et al., 1985, Jordan, 1986], which used to be state of the art in machine translation and text classification tasks such as sentiment analysis.

Here, the last state of the network is fed into the network alongside the rest of the input. For machine translation, models are often split into encoders and decoders. This serves the purpose that the encoder creates some internal abstract representation of the word sequence, which is then decoded into, e.g., a different language by the decoder. Thus, the encoder can process one word after another, taking its former output and the next as input. Once the encoder has processed all the words in the input sentence, the resulting representation is fed into the decoder. The latter creates one output word after the other, starting with the encoded representation, and then takes the last output word alongside the encoded representation as input to produce the output sentence.

How words are represented in a numeric form is crucial here, as different word representations contain varying amounts of information and thus play a role in the performance of a model. Word embeddings have been adopted as the standard here, as they are utilized in all state-of-the-art text processing models.

2.1 Word Embeddings

Word embeddings were first researched on by Firth [1957] and Bengio et al. [2000].

In such an embedding, a multi-dimensional hyper-space is created into which words are mapped as vectors. The aim here is to map words that occur in similar contexts and frequency to similar positions within the vector space so that words that are semantically similar lie in similar positions. Hence, this approach encodes more information than a plain one-hot encoding, i.e., a vector the size of the word corpus, where each component corresponds with one word from the latter, containing only zeros except for a one in the position that represents the word at hand. Then the semantic similarity of

words can be computed using specific distance measures, e.g., cosine similarity, where the cosine of the angle between the two-word vectors is calculated [Singhal et al., 2001]. To create a word embedding, a matrix M_e is created, comprised of N_c columns and N_d rows, where N_c is the number of words in the vocabulary, and N_d is the number of dimensions in the embedding. The matrix is used as an embedding layer of a neural network [Rosenblatt, 1960, Werbos, 1994], so that it can be trained alongside the rest of it using the backpropagation algorithm [Werbos, 1994, Bengio et al., 2000].

Henceforth, word embeddings play an essential part in models that rely on encoding the meaning of words into numerical representations. Amongst these, attention-based networks, above all the Transformer model, have recently gained popularity.

2.2 The Transformer Architecture

Language modelling has been revolutionized through the introduction of the Transformer architecture, first proposed by Vaswani et al. [2017]. It is designed to have a non-recursive structure that takes in a sequence of tokens, processing the entire segment simultaneously. The original model proposed by [Vaswani et al., 2017] contains an encoder and a decoder for translation tasks. Therefore, word tokens are embedded in a word embedding to encode semantic similarity between words [Firth, 1957, Bengio et al., 2000]. Secondly, a positional encoding of the sentence is added to the resulting vector, composed of learned or fixed parameters that add information about the order of words in the sentence.

As a result, a series of embedded tokens are fed into the Transformer encoder, which creates an abstract representation of the sequence that a Transformer decoder can, in the end, decode. The mechanism that sets this class of models apart from other techniques and which allows Transformers to process whole sequences at once is called attention.

2.2.1 Attention

The Transformer model proposed by [Vaswani et al., 2017] uses attention in three distinct

ways: self-attention, encoder-decoder attention and multi-head attention. Each of these variants relies on computing semantic similarity between words in the input sequence with each other or words from the output sequence. With this, the target is that the model eventually learns firstly which words in the input sentence are essential to what other words are in it, e.g., that in the sentence "The girl's mother congratulated her", the words "girl" and "her" correlate in meaning. This type of attention is called self-attention. Secondly, the model is supposed to learn how words in the input sequence relate to words in the output sequence, e.g., that between the sentences "I love you" and "Je t'aime", the words "love" and "aime" are strongly semantically correlated. In turn, this second variant is known as encoder-decoder attention. Lastly, multi-head attention is used in both of the described attention techniques. Here, multiple attention heads are used alongside each other such that different attention patterns can be learned for the same sequence.

To achieve this, three matrices Q , K and V are created that denote the queries, keys and values. Each column in Q corresponds with a word in the input sequence for self-attention or the output sequence for encoder-decoder attention. In contrast, the columns in K and V correspond with tokens in the input sequence. Q , K and V are computed using three learned matrices W_q , W_v and W_k to project the input embedding linearly.

In the end, the attention vector is computed by multiplying Q and K , scaling by $\frac{1}{\sqrt{d_k}}$, i.e., the inverse of the square root of the number of dimensions in K , taking the softmax, and lastly, multiply the resulting matrix by V .

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

The result of this is a matrix with N_i columns, with i being the input sequence length, and N_o rows, with o being the output sequence length for encoder-decoder attention and the input sequence length for self-attention. In a trained Transformer, this matrix will show the learned semantic correlation between the tokens. Another way of understanding this is that both sequences that attention is computed between are multiplied by attention weights, which, over the training,

learn how inputs correlate. Subsequently, the two resulting projections are multiplied and fed into the softmax function, creating a probability distribution over the computed correlation. Lastly, this distribution is multiplied with another linearly projected version of the input vector, which creates a matrix with attention values for each pair of tokens in the two input vectors [Bahdanau et al., 2014, Vaswani et al., 2017].

Using the attention mechanism, the results obtained by [Vaswani et al., 2017] surpassed the ones acquired with RNNs, namely in text translation. At the same time, training went down significantly for Transformer models as operations are parallelizable to a far greater extent, such that training on GPUs that can handle many simultaneous calculations was much faster.

2.3 The Vision Transformer

The field of Computer Vision has adopted Convolutional Neural Networks (CNNs) as its standard for almost all image processing tasks [LeCun et al., 1999]. These use convolution to process images, sliding filters of a fixed size across an image to extract particular features such as colour and edges. On the other hand, Dosovitskiy et al. [2020] proposed a transformer model for image processing (Vision Transformer, ViT), using pure attention mechanisms to replace convolutions. To create an embedding for the input images to be handed to the transformer, they are split into non-overlapping patches of size 16x16 pixels. Subsequently, the patches are flattened and linearly projected. Lastly, the input embedding is fed into a Transformer encoder, at the end of which is a Multi-Layer Perceptron (MLP) head [Rosenblatt, 1960, Werbos, 1994] used to classify images, i.e., a traditional neural network with fully connected layers.

The ViT shows comparable or better performance over CNNs on several data sets. However, Dosovitskiy et al. [2020] pre-trained their model on a non-public data set containing over 300mil. pictures. Only after this pre-training step, the model had generalized enough visual knowledge to perform comparably to CNNs.

2.4 Contemporary Research in Multi-Modal Processing

In a visual question-answering task, Hu and Singh [2021] proposed a multi-modal model that takes text and an image as input, with text as output. While the text is processed using a standard transformer in this case, the image is first converted by a convolutional network as the pre-processor. The textual and visual embedding are adjoined and sent to a final transformer in a multi-head output setting. The researchers wanted to train the model on different tasks for different output heads of the versatile model. The model is able to do image classification, object detection, and VQA, among others. Similar approaches have been taken by Tsai et al. [2019], Yu et al. [2019], Huang et al. [2020], and Prakash et al. [2021].

Hence, contemporary multi-modal networks never use a single transformer encoder that two modalities are presented to at the same time. This aspect draws a clear gap in the current state of the research.

2.5 Research Goal

Transformers have proven to be powerful in both the textual and image domain [Vaswani et al., 2017, Dosovitskiy et al., 2020]. Although there may still be challenges regarding the inductive bias associated with patch extraction, performing cross-attention between text and image patch tokens could be potent in multi-modal processing. Therefore, a sensible next step would be a multi-modal architecture that takes as input text and an image, processing those with a single Transformer module. To see whether convolution is important for reliably extracting image features, a pure Transformer-based variant will be compared to a variant complemented with a convolutional pre-processor.

3 Methods

3.1 Task design: four-quadrant sequence classification

To realize a mixed visual/sequential task, a choice must be made regarding the primary target object, and a solution for the sequential classification task for a series of such objects needs to be

found. We choose to use the individual digits from the MNIST [LeCun et al.] and clothes from the Fashion MNIST [Xiao et al., 2017] data sets as the aforementioned basic objects. For the serialization of the tasks, the individual images are pasted together in a single image with four quadrants. The task for the system then is to classify a sequence of objects in the regular reading order, starting with the top left, then the top right, followed by the bottom left, and finally, the image in the bottom-right quadrant. The target label for the output will either be consistent or inconsistent with the object sequence. The task is henceforth to decide whether the given image sequence corresponds to the target text string of four labels. Figure 3.1 shows a sample data point for the task design using (images of) capital letters.

Furthermore, to help the network learn faster and more reliably, alongside the two output units for the true/false classification, one output unit is added for each class (e.g., 10 for MNIST, creating 12 output units in total). Hence, the eventual target will contain a true/false assertion and a vector denoting which classes the image contains. This dual-task setup also allows checking whether the individual object classification is working and thus aids in the explainability.

Image		Text	Target
A	B	A B C D	True
C	D		

Table 3.1: A prototypical data point for the task design. Left: an image made up of four sub-figures (A,B,C and D). Middle: the output sequence. Right: a target decision denoting whether the image and text sequence logically fit.

3.2 Data Sets

The proposed model will be trained and tested on two data sets, composed of quadrant images derived from the individual samples from MNIST and the Fashion MNIST data set, i.e., 2x2 such sub-images

are combined into a single image for training or testing.

3.2.1 The MNIST Data Set

In order to train the model in question, the MNIST data set [LeCun et al.] will be used, which contains handwritten digits from zero to nine. The original data set contains 60,000 training and 10,000 test data points. Each data point contains a 28×28 pixel image of a handwritten digit and a label, precisely a number from zero to nine shown on the image.

The data set will be pre-processed as shown in table 3.1, creating four-quadrant images of handwritten digits together with a number sequence and a true/false assertion. Table 3.2 shows a sample data point from the pre-processed data set.

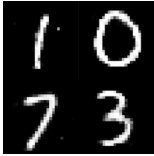
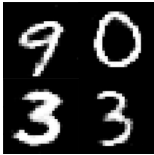
Image	Numbers	Target
	1 0 7 3	True
	9 1 3 4	False

Table 3.2: Two sample images and their corresponding number sequences and targets from the pre-processed data set.

3.2.2 The Fashion MNIST Data Set

Additionally, the model will be trained and tested on the Fashion MNIST data set [Xiao et al., 2017], which contains 60,000 training and validation as well as 10,000 test pictures of black and white fashion products from the Zalando online store. Henceforth, all meta attributes of the data set are the same as for the MNIST data set: the size, the image dimensions, and the number of classes. On the other hand, the only difference is that the images show clothing items instead of handwritten digits. All pre-processing steps for this data set shall be

identical to the ones employed for the MNIST data set, such that four images are concatenated and presented together with a true or false sequence that the model will have to classify. Table 3.3 shows a sample data point from the data set.


Image	Numbers	Target
	Shoe Shirt Shirt Jacket	False

Table 3.3: A sample datapoint from the pre-processed Fashion MNIST data set, showing a picture, its corresponding sequence of clothing items, and the target.

3.3 Model Architecture

Figure 3.1 shows the model architecture proposed. It orients itself closely at the work by Vaswani et al. [2017] and Dosovitskiy et al. [2020].

At the beginning of the input stream, both modalities are embedded using separate modules. On the one hand, the text sequence is embedded using a standard word embedding. On the other hand, the image embedding differs between the convolutional and non-convolutional conditions. In the former, the image is first presented to a **convolutional pre-processor** before being further processed by a **patch embedding layer**. In the non-convolutional condition, only the last step is performed. The reader is referred to table C.1 for a detailed listing of the input and output dimensions within each layer of this visual processing stream.

3.3.1 Convolutional Pre-Processor

In the convolutional condition, the image is fed through N_c convolutional layers with 3×3 sized kernels each, where the i th layer has 16^i channels (e.g., the first layer has 16 channels, the second one has 32 channels, etc). The design of this convolutional pre-processor is inspired by the VGG-16 model developed by Simonyan and Zisserman [2014], who employed layers of doubling channel numbers. VGG-16 is a suitable candidate

for a pre-processor as it is one of the more basic CNNs that are relatively shallow compared to other architectures, yet it still performs quite well. Therefore, since it is even more simplified in the current research, it does not add immense complexity to the model. Hence, a fair comparison between the two conditions is still possible.

3.3.2 Patch Embedding

Subsequently, the (convoluted) image is embedded by partitioning it into non-overlapping patches of 4×4 pixels. After that, the resulting patches are flattened, transforming the x , y and *channel* dimension into a single one. For the last step of the image embedding, the resulting vectors are linearly projected.

The word and the patch embedding modules linearly project the tokens onto a latent space of the same dimensionality. Usually, 512 or 768 is chosen as the number of dimensions. However, the exact number of dimensions will be a hyper-parameter optimized for the model at hand (as described in section 3.4.3).

3.3.3 Classic Transformer Input Stream

Subsequently, the embedded words, patches and a learnable classification token are concatenated and positional encoding is added, as done in the work by Dosovitskiy et al. [2020].

The eventual sequence is fed through a standard transformer encoder of N_e layers, at the end of which the classification token is classified using a feed-forward layer with exactly eleven output neurons, one for the actual true/false classification, and ten neurons for each class in the respective data set. The latter aims to help the model develop an understanding of the data set, its classes, and the pictures. Lastly, a sigmoid activation is placed at the end of the model.

3.3.4 Why Just a Single Encoder?

In other studies on multi-modal, researchers always presented the two modalities to two separate encoders (as seen in section 2.4). This research gap is closed in this study to check whether self-attention proves to be powerful between modalities.

As seen in the work by [Vaswani et al., 2017], their transformer model learns to pay attention to specific words within a sentence concerning words it encounters. Specifically, when attention is calculated between two sentences, each word in the first sentence attends to each word in the second one so that the model can understand which words correlate in meaning or importance to the current task. The exact mechanism appeared in the research published by [Dosovitskiy et al., 2020], where different image parts were essential for processing other image parts. The attention maps in their study show how the model learns to pay attention to some parts of the image, e.g., outlining the exact boundaries of an airplane while not attending to the rest.

Henceforth, the model architecture in question should combine these two findings. Specifically, in the self-attention layers, the model should learn which parts of the text are essential for certain parts of the image and vice versa. Therefore, it should still maintain the ability to correlate words with words and image parts with other images. The current model design can achieve this as both modalities are concatenated. The resulting attention matrix maps from each information token (image and text) to every other token.

3.4 Training and Evaluation

3.4.1 Pre-Processing

The data set is split into a training set of 55,000, a validation set of 5,000, and a test set of 10,000 data points. During training, the four-quadrant images are sampled live from the full training set and the true/false sequence labelling was also determined live, per I/O pair. In particular, an image is complemented with three additional randomly sampled images from the data set. Furthermore, a false sequence is shown with a 50% likelihood. In such cases, at least one of the four tokens is replaced with a randomly sampled one from the remaining classes. The same replacement procedure is done to the other three numbers with a likelihood of 25% each. To test whether all labels occur uniformly in each quadrant, a distribution was computed during the experiment runs, which can be seen in table D.1.

In the end, this pre-processing step substantially

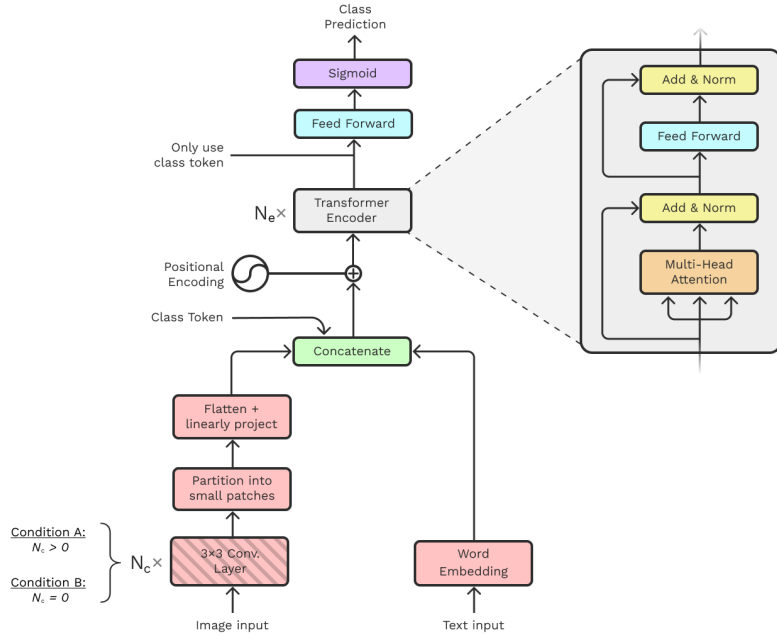


Figure 3.1: The proposed model architecture. Both modalities are embedded, concatenated, complemented with a class token and positional embedding, and fed into a transformer encoder. The resulting output of the class token is used for the classification task. If the number of hidden units in the visual pathway N_c is larger than zero (condition A), the left branch is a CNN with exactly that many 3×3 kernels and $16i$ channels ($16, 32, 48, \dots$). If the number of hidden units N_c equals zero, no convolutions are used, and the model reduces to an end-to-end transformer with crude input patches (4×4 pixels) from the input of 56×56 pixels (condition B). A detailed listing of the input and output dimensions of each layer of the visual pre-processing stream can be found in table C.1.

increases the number of unique data points. Four images are taken out of a set of 60,000 images, and the corresponding text sequence can be true or false. Additionally, in the false case there is $N_{\text{classes}} - 1 = 9$ possible values for each token. In consequence, the resulting number of unique data points in the set is $60,000^4 \cdot (9^4 + 1) \approx 8.5 \times 10^{22}$. However, in the eventual experiment, only a fraction of these data points will be shown, depending on how long the experiment runs.

3.4.2 Loss Metric and Optimizer

Binary Cross Entropy with Logits is chosen as the loss function, a standard metric for problems in which binary patterns have to be learned [Goodfellow et al., 2016]. This loss function combines a

sigmoid layer with Binary Cross Entropy. Because the output the model is trained on has multiple binary patterns, i.e., a true/false assertion and multiple classes present in the image, loss functions such as Categorical Cross Entropy do not work here.

For a model output vector y and a target vector \hat{y} , the formula for Binary Cross Entropy with Logits is composed as follows:

$$L(y, \hat{y}) = \begin{bmatrix} l_1 \\ \dots \\ l_N \end{bmatrix}$$

$$l_n = \hat{y}_n \cdot \log \sigma(y_n) + (1 - \hat{y}_n) \cdot \log(1 - \sigma(y_n))$$

with N being the number of components of the output vector \hat{y} , y_n being the n th component of

the target vector, and \hat{y}_n being the n th component of the output vector, i.e., 0 or 1. In addition, σ refers to the sigmoid function, i.e., $\sigma(z) = \frac{1}{1+e^{-z}}$ with z being the input number.

Furthermore, the ADAM optimizer is used, which is a variant of Stochastic Gradient Descent, converging to minima much faster and being more likely to find lower minima as it makes use of momentum techniques [Kingma and Ba, 2014].

3.4.3 Experiment Setup

Using these ingredients, two experiments are set up, one with a model that uses a convolutional pre-processor for processing images and another one that relies solely on transformer-based attention. The model architecture, as well as the training, validation, and test procedures, are implemented using the PyTorch toolbox [Paszke et al., 2019].

Specifically, a validation scheme is employed where the model is tested on the validation set after each epoch to monitor the validation loss. Furthermore, this allows for the training to be stopped using an early stopping technique that quits the training if the validation loss does not decrease for three epochs.

Additionally, each experiment conducts a hyperparameter optimization, performing a random grid search on sampled values for the dropout, learning rate, the number of convolutional layers in the pre-processor, the number of transformer encoder layers, the embedding dimension of the word and patch embeddings, and the number of attention heads in the transformer encoder layers. Each hyperparameter setting is run five times, and the accuracy’s resulting mean and standard deviation are calculated to account for differences due to the random initialization of model parameters.

Both experiments are run on a single NVIDIA V100 GPU node for a maximum of 54 days of GPU time.

Lastly, to draw definitive conclusions about the differences between the two conditions, two unpaired t-tests [Student, 1908] will be conducted, one for each experiment.

3.4.4 Accounting for Comparability

Henceforth, to sum up, it is made sure that the results of both experiments will be comparable for two reasons.

Firstly, both models rely on splitting the input image into patches of 4x4 and using the same embedding dimension as a bottleneck. Consequently, the convolutional pre-processor cannot already transform the image patches into extensive feature vectors because those vectors are split and projected onto a smaller hyperspace.

Furthermore, both experiments are limited by a maximum GPU time of 54 days so that no experiment can optimize longer than the other.

4 Results

The goal of the current research was to propose a novel model architecture that simultaneously processes visual and textual data using a single Transformer encoder. Furthermore, the model was trained and tested on two data sets, i.e., the MNIST and the Fashion MNIST data set, across two conditions, one where it contained convolutional elements and one where it did not. A custom task design was implemented in which four images of the respective data set were shown in a grid-like fashion, together with a sequence of tokens. The model had to decide whether it was correct or false regarding the reading order of the objects in the figure.

In each experiment run, about 6.4mil. randomly sampled unique data points were used to train the proposed architecture before the early stopping mechanism quit the training. Table 4.1 shows the results attained in the experiments. On the MNIST data set, when coupled with a convolutional backbone, the proposed architecture achieved a mean accuracy of 71.15% with a standard deviation of 0.001%. On the same data set, without convolutional elements, the model scored a mean accuracy of 70.20% with a standard deviation of 0.007%.

In addition, on the Fashion MNIST data set and combined with a convolutional backbone, the model attained a mean accuracy of 71.03% with a standard deviation of 0.001%. On the same data set, without the convolutional backbone, the

model scored a mean accuracy of 70.56% with a standard deviation of 0.002%.

	MNIST	Fashion MNIST
Convolutional Image Embeddings	97.73 ±0.5%	95.70 ±0.3%
Disjoint Image Patch Embeddings	92.14 ±0.5%	92.39 ±0.2%

Table 4.1: The accuracy obtained on the True/False task for the two data sets. The rows denote the two conditions: (1) where the image embedding contained convolutional features and (2) a disjoint image-patch sequence obtained from four 28x28 pixel input images. The means and standard deviations were computed over five different runs each, and in each run, the model was trained on about 6.4mil. randomly generated unique image/sequence pairs.

To find out whether the differences across the two conditions are statistically significant, two unpaired two-sided t-tests were performed. For the MNIST data set, there was a highly significant difference in the mean accuracy scores between the convolutional ($M=97.73$, $SD=0.5$) and non-convolutional ($M=92.14$, $SD=0.5$) condition; $t(8)=17.6771$, $p < 0.0001$. Additionally, for the Fashion MNIST data set, there was a highly significant difference in the mean accuracy between the convolutional ($M=95.70$, $SD=0.3$) and non-convolutional ($M=92.39$, $SD=0.2$) condition as well; $t(8)=20.5277$, $p < 0.0001$.

Moreover, Figures 4.1a-4.1d show the loss and accuracy curves of the four best performing models just mentioned.

5 Discussion

This study tested a novel unified Transformer model that processes images and text with a single encoder module in two conditions. In the first condition, the model processed images using a convolutional pre-processor. In the other condition, the entire model relied on Transformer-based attention, leaving out convolutional elements. The model was further trained and tested on two data sets: the MNIST data set of handwritten digits and the Fashion MNIST data set that features clothing images categorized into ten classes. Both data sets were modified such that datapoints contained four concatenated images and a true or false corresponding text sequence so that the model could learn to discriminate true from false image sequence pairs in regards to the reading order of the objects in the figure.

The results showed that the convolution-based model performed significantly better than the non-convolutional model in both experiments. Namely, the accuracy score was 5.5% higher for the MNIST data set (97.73% vs. 92.14%) and 3.3% higher for the Fashion MNIST data set (95.70% vs. 92.39%).

5.1 Interpretation of Results

The convolutional condition’s accuracy scores were slightly yet significantly higher than those obtained in the non-convolutional condition (about 3-6%). These results show that both architectures can perform similarly well in a setting with simple images, such as one-channeled images of handwritten digits between zero and nine.

On the other hand, the significant differences in accuracy give evidence for the hypothesis that convolution-based pre-processing aids image processing in Transformer-based models. It can be argued that convolutional networks, using a stride of 1 pixel in the horizontal and vertical direction, present less of an inductive bias than a visual transformer where a chosen ‘reading order’ and patch size are imposed on a given image. Not that this should be problematic: Human-eye movements also scan pictures in a stereotypical manner, attending to points of fixation in a particular order [Salvucci and Anderson, 1998, Noton and Stark, 1971, Reisberg, 2015].

Only when considering fields of high correlation

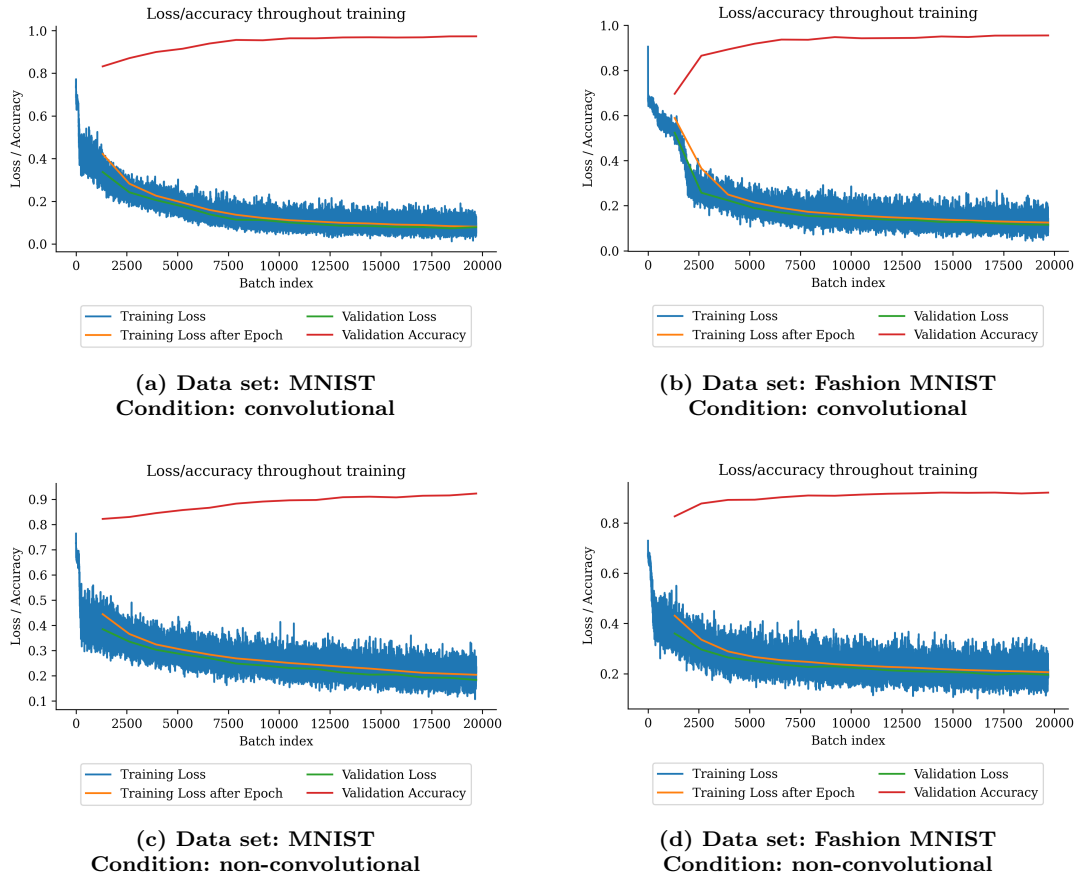


Figure 4.1: The loss/accuracy curves of the best performing model trained on each data set and condition.

over a (potentially long) patch sequence do transformers lack an inductive bias, making them very suitable for learning various sequential tasks. ViTs, on the other hand, rely on splitting images into non-overlapping patches. This is necessary because performing attention between every pair of pixels in an image would be computationally infeasible due to quadratic time and space complexity. Thus, ViTs cannot correlate pixels within patches but only between patches. A potential downside of this is that, e.g., an image patch showing a chair would have a different encoding than one where the chair is shifted to the right by just one pixel. In contrast, CNNs process each pixel, correlating it with only the pixels surrounding it. Furthermore, when breaking down images into non-overlapping patches, the image is likely broken in essential re-

gions. For example, an MNIST image of the number "4" could be broken at the point where the lines cross so that the digit is not clear from the resulting representation anymore.

As a consequence of the three mentioned problems, ViTs need to be trained on a considerably higher amount of data to enable them to generalize. Firstly, they take longer to understand that in images, pixels around a pixel matter much more than pixels on the other side of the picture. Secondly, it takes a lot longer for the model to learn a patch embedding that maps similar objects to very close points in the latent space, even if they are shifted by a number of pixels. These problems and the resulting implication are also why Dosovitskiy et al. [2020] trained their ViT on a data set containing over 300 million images. At the same time,

CNNs can achieve similar performance on considerably smaller data sets [Chollet, 2017, Krizhevsky et al., 2012, Szegedy et al., 2014]. In this study, image patches had a size of 4x4 black and white pixels, which made it easier for the model to learn to embed patches properly, but with complex three channelled images and 16x16 pixel patches, the amount of data and training time needed would have been a lot higher. CNNs do not have this problem, which is a tremendous advantage as smaller companies and research institutions cannot satisfy the requirements to successfully train a ViT on such a large amount of data.

In the early days of neural-network-based computer vision, researchers found that CNNs do a much better job at extracting features from an image than regular fully connected MLPs [LeCun et al., 1998]. In the same way, using a fully-connected dense layer to map image patches into some latent space that encodes features may just not be optimal. The results of this study show that ViTs still possess a fundamental design flaw that they would need to overcome to leave behind the aforementioned problems and become a true competitor to CNNs.

5.2 Strengths, Limitations, and Prospect on Further Research

The current study proposes the first-ever model architecture that processes image and text simultaneously using a single Transformer encoder. Consequently, the strength of this work is that it takes the most recent status quo further, combining findings from very influential papers [Vaswani et al., 2017, Dosovitskiy et al., 2020] to create a new, more general Transformer architecture.

Furthermore, this study clears up the myth that ViTs are a more general, superior architecture compared to CNNs [Dosovitskiy et al., 2020, Bai et al., 2021, Han et al., 2022]. There are some obvious challenges associated with the processing stream of ViT. Only if these shortcomings were overcome would it offer a suitable and possibly more general alternative to convolution.

Nevertheless, the MNIST and Fashion MNIST data sets remain highly limited since images are one-channelled and only show a single object. Additionally, both data sets have only ten classes. The results of this work cannot be fully generalized to all

kinds of images, as real-world images would contain multiple objects and sophisticated color information. However, as the goal of this study was a proof-of-concept of a multi-modal transformer architecture, training and testing the proposed architecture on more elaborate data sets is left for future studies.

Therefore, the next step for further research would be to test the model proposed in this paper on more extensive and complex data sets. In particular, the Coco [Lin et al., 2015] and Visual Genome [Krishna et al., 2017] would serve as promising candidates here. In the former, data points contain an image and multiple correlated captions so that the model could be trained to distinguish between true and false captions, as in this study. On the other hand, in the Visual Genome data set, datapoints contain an image and associated question-answer pairs. Hence, the model could be trained to answer questions based on the images. Based on the results of this research, a possible hypothesis here would be that the convolutional variant will again perform better. Furthermore, as the images in these data sets are a lot more complex than in the MNIST and Fashion MNIST data sets, it could also be hypothesized that the difference in accuracy between the convolutional and non-convolutional settings will be even higher than in this study.

In addition, it is likely that, as in the research by Dosovitskiy et al. [2020], the data sets needed have to be a lot larger for the non-convolutional architecture to succeed in generalizing knowledge.

Moreover, further research could conduct experiments on similar architectures that consider even more modalities. For example, the findings of this study could be used to produce a reinforcement learning agent that perceives auditory, visual, and haptic information and processes it using a single Transformer module. The generality of the attention mechanism could thus potentially allow other models to learn any task and process any information, bringing us one step further towards general intelligence.

References

- Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

- Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
- Yutong Bai, Jieru Mei, Alan L Yuille, and Cihang Xie. Are transformers more robust than cnns? *Advances in Neural Information Processing Systems*, 34:26831–26843, 2021.
- Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. *Advances in Neural Information Processing Systems*, 13, 2000.
- François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1251–1258, 2017.
- Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- John R Firth. A synopsis of linguistic theory, 1930-1955. *Studies in linguistic analysis*, 1957.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
- Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence*, 2022.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.
- Ronghang Hu and Amanpreet Singh. Unit: Multimodal multitask learning with a unified transformer. *arXiv preprint arXiv:2102.10772*, 2021.
- Jian Huang, Jianhua Tao, Bin Liu, Zheng Lian, and Mingyue Niu. Multimodal transformer fusion for continuous emotion recognition. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3507–3511. IEEE, 2020.
- MI Jordan. Serial order: a parallel distributed processing approach. technical report, june 1985-march 1986. Technical report, California Univ., San Diego, La Jolla (USA). Inst. for Cognitive Science, 1986.
- Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. URL <https://arxiv.org/abs/1412.6980>.
- Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123(1):32–73, 2017.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.
- Yann LeCun, Corinna Cortes, and Christopher J.C. Burges. The mnist database of handwritten digits. URL <http://yann.lecun.com/exdb/mnist/>.
- Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision*, pages 319–345. Springer, 1999.
- Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

- David Noton and Lawrence Stark. Eye movements and visual perception. *Scientific American*, 224(6):34–43, 1971.
- Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL <http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.
- Aditya Prakash, Kashyap Chitta, and Andreas Geiger. Multi-modal fusion transformer for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7077–7087, 2021.
- Daniel Reisberg. *Cognition: Exploring the science of the mind: Sixth international student edition*. WW Norton & Company, 2015.
- Frank Rosenblatt. Perceptron simulation experiments. *Proceedings of the IRE*, 48(3):301–309, 1960.
- David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.
- Dario D Salvucci and John R Anderson. Tracing eye movement protocols with cognitive process models. In *Proceedings of the twentieth annual conference of the cognitive science society*, pages 923–928. Routledge, 1998.
- Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- Amit Singhal et al. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4): 35–43, 2001.
- Student. The probable error of a mean. *Biometrika*, pages 1–25, 1908.
- Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014. URL <https://arxiv.org/abs/1409.4842>.
- Yao-Hung Hubert Tsai, Shaojie Bai, Paul Pu Liang, J Zico Kolter, Louis-Philippe Morency, and Ruslan Salakhutdinov. Multimodal transformer for unaligned multimodal language sequences. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2019, page 6558. NIH Public Access, 2019.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.
- Paul John Werbos. *The roots of backpropagation: from ordered derivatives to neural networks and political forecasting*, volume 1. John Wiley & Sons, 1994.
- Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017. URL <http://arxiv.org/abs/1708.07747>.
- Jun Yu, Jing Li, Zhou Yu, and Qingming Huang. Multimodal transformer with multi-view visual representation for image captioning. *IEEE transactions on circuits and systems for video technology*, 30(12):4467–4480, 2019.

A Multi-Layer Perceptrons

The multilayer perceptron (MLP) was first proposed by Rosenblatt [1960] and later developed further by Werbos [1994]. Being the first so-called Neural Network (NN) and thereby one of the most influential and famous models in supervised learning, it can be used as a universal function approximator. It is constructed of multiple connected layers that are in turn composed of so-called neurons. In particular, neurons take numbers as input and compute a weighted sum of those before adding a bias and applying an activation function, specifically a non-linearity. Classically, the sigmoid-function $s(z) = \frac{1}{1+e^{-z}}$, with z being the input number, is used as this non-linearity. However, other functions with different advantageous properties have been proposed, e.g., the rectifier linear unit function, $ReLU(x) = \max(0, z)$ with z being the input number [Agarap, 2018]. In the case of the classic MLP, also denoted as the Fully-connected Feed Forward Neural Network, the neurons present in one layer all connect to every neuron in the next layer, as pictured in figure A.1.

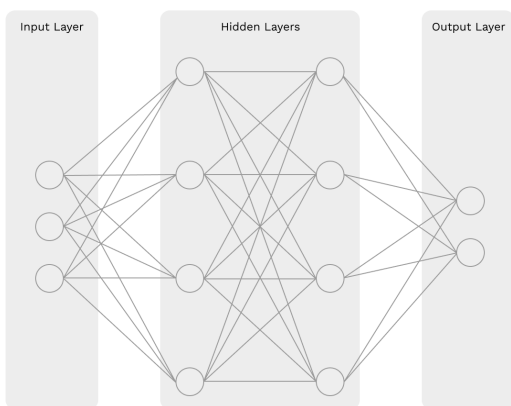


Figure A.1: A schematic of a Fully-connected Feed Forward Neural Network.

The mathematical mechanics just described can thus also be imagined as each branch connecting from one neuron to the next having an associated connection strength or weight multiplied by each number that passes through that path. Each node or neuron in the network takes all the incoming weighted numbers, sums them up, and applies the non-linearity. After that, the neuron passes the

resulting output through each connecting path that branches off it. Figure A.2 graphically depicts this mechanism.

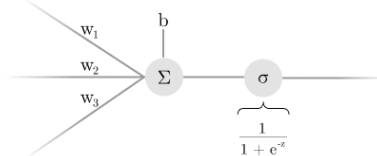


Figure A.2: A schematic of the workings in an artificial neuron. Weights are denoted as w_i , the bias as b , Σ refers to the summation operation, and σ represents the non-linearity.

The equation of a single neuron is hence comprised as follows:

$$z(\vec{x}) = \sigma(\vec{w} \cdot \vec{x} + b)$$

where \vec{x} is the input vector, \vec{w} is the weight vector, b is the bias and σ is the non-linear activation function.

MLPs are inspired by the mechanics of the brain, in which neurons receive input from connecting neurons and fire if a certain threshold is reached. In an MLP, the connection strength between neurons is modelled as the weights w , the threshold is represented as the bias b , and the non-linearity σ is used to model the firing behaviour. However, training the network would not be directly possible with discrete firing behaviour. Therefore, an activation function with a continuous derivative is used so that the backpropagation algorithm can be used, which relies on computing the gradient of the network function.

A.1 The Backpropagation Algorithm

To train an MLP, Werbos [1994] describes that a loss function L is computed between the model output and the desired output, for instance, the mean squared error (MSE):

$$L_{MSE}(Y_i, \hat{Y}_i) = \frac{1}{n} \sum_{i=0}^n (Y_i - \hat{Y}_i)^2$$

where n is the number of points in the data vector, Y_i is the target value of a data point, and \hat{Y}_i is the model's estimation.

Subsequently, the Backpropagation algorithm is utilized, computing the gradient of this loss-function ∇L , hence the vector of partial derivatives of the loss function with respect to the weights and biases. Subsequently, gradient descent is applied so that the cost function converges to a local minimum, creating an optimal model. Thereby, all the weights are updated based on the gradient:

$$\vec{w} \leftarrow \vec{w} - \alpha \nabla L$$

where \vec{w} is the vector of weights and biases in the model, α is the learning rate (a hyperparameter) and ∇L is the gradient of the loss function L in relation to the weights w and biases b of the network.

B Convolutional Neural Networks

Convolutional Neural Networks (CNNs) were pioneered by [LeCun et al., 1998] (1998) as a different way of processing information in an NN that is tailored to visual data. In pictures, pixels at one edge often do not semantically correlate with the ones on the other, as they can be part of very different objects captured in the same image. Instead, the area surrounding a particular pixel is much more meaningful to what it refers to. Henceforth, [LeCun et al., 1998] utilized this insight, resorting to the convolution operation to encode the correlation of pixels to nearby other pixels.

An n -dimensional discrete signal, e.g., an audio signal or an image, can be convoluted using a filter of the same dimensionality but smaller (or equal) size. For the case of an image, the filter is slid along it, and the dot product between the pixels it covers and itself is computed. As a result, a new, smaller image is created that contains information about the correlation between neighbouring pixels, as shown in figure B.1.



Figure B.1: A depiction of a two dimensional convolution. The dark grey area represents a 3x3 filter that slides over the pixels, computing the dot product between the overlaid area and the filter and assigning it as a pixel to the resulting image.

Convolutions can extract various features from images, including edges and colour. [LeCun et al., 1998] made use of convolutional filters instead of fully connected neurons in their NN to let the first layers of their network extract, e.g., edges, while the later layers combined these edges into shapes so that the network could, in the end, recognize objects. In that case, the numbers associated with the filters denoted the weights that the network learned.

The researchers combined these convolutional

layers with pooling layers, which reduce the dimensionality of an image by taking the maximum, minimum or average value out of each $n \times n$ (e.g., 2×2) patch, summarizing the information in small regions of the feature map. Thence, the number of weights needed to learn features and thus be able to recognize objects in images was drastically reduced in comparison to using regular fully connected neural networks, as convolutional filters have tremendously fewer weights than fully connected layers and pooling layers dramatically reduce the number of weights again.

Nevertheless, compared to fully connected MLPs, [LeCun et al., 1998] still achieved much higher accuracy on, e.g., the MNIST data set that contains handwritten digits. Being much easier to train and performant than regular NNs, CNNs were quickly adopted as the state-of-the-art in image processing. Many variants of the original structure have been proposed to achieve even higher accuracy, including AlexNet [Krizhevsky et al., 2012], ResNet [He et al., 2016] and Xception [Chollet, 2017].

C Output Dimensions within Visual Pre-Processing Stream

	Pattern	<i>Example: no convolution</i>	<i>Example: 4 conv. layers</i>
Original	$1 \times 56 \times 56$	$1 \times 56 \times 56$	$1 \times 56 \times 56$
Conv. Layers	$\begin{aligned} &\max(1, 16N_c) \\ &\times \\ &56(56 - 2N_c) \\ &\times \\ &56(56 - 2N_c) \end{aligned}$	/	$\begin{aligned} &16 \times 54 \times 54 \\ &32 \times 52 \times 52 \\ &48 \times 50 \times 50 \\ &64 \times 48 \times 48 \end{aligned}$
Patcherize	$(\frac{56-2N_c}{4})^2 \times \max(1, 16N_c) \times 4 \times 4$	$196 \times 1 \times 4 \times 4$	$144 \times 64 \times 4 \times 4$
Flatten	$(\frac{56-2N_c}{4})^2 \times 16\max(1, 16N_c)$	196×16	144×1024
Lin. Proj.	$(\frac{56-2N_c}{4})^2 \times D_e$	$196 \times D_e$	$144 \times D_e$

Table C.1: The respective output dimensions of each layer in the image pre-processing stream of the network, together with two examples, one without any convolution, and another one with four convolutional layers. N_c indicates the number of convolutional layers, while D_e denotes the embedding dimension. The dimensions shown in the last row depict the length of the sequence that will be concatenated with the embedded text sequence, together with the size of each token vector.

D Distributions of Class Labels within Quadrants

Class Label	Q1	Q2	Q3	Q4
0	2.47%	2.45%	2.44%	2.46%
1	2.82%	2.80%	2.82%	2.84%
2	2.46%	2.47%	2.50%	2.48%
3	2.52%	2.58%	2.58%	2.56%
4	2.46%	2.45%	2.43%	2.43%
5	2.26%	2.29%	2.26%	2.24%
6	2.49%	2.46%	2.46%	2.47%
7	2.61%	2.62%	2.59%	2.60%
8	2.45%	2.42%	2.45%	2.42%
9	2.46%	2.46%	2.47%	2.49%

Table D.1: The distributions of each class within the four quadrants of the generated figure for the MNIST and Fashion MNIST data sets.