CV IS Study

Maxwell Tang, Donald Winkelman

18 Dec 2024

1 Structural Overview

This paper is the combination of an individual study of Mechanistic Interpretability (MI) by Donald Winkelman and a joint-study of applying Mamba to OCR by Maxwell Tang and Donald Winkelman.

The first two sections of this paper cover two self-contained MI studies. The first section applies feature visualization to VGG16, a popular image classification model, while the second utilizes the integrated gradient method to interpret a sentiment analysis transformer.

The Mamba OCR study spans sections 3-7 and explores Mamba's ability to perform in-context learning on various OCR tasks. Mamba is a state space model (SSM) that, through various modifications, is able to effectively hold onto long-range context while remaining O(n) in the number of tokens being processed.

All three studies share a bibliography and appendix which can be found at the bottom.

2 Computer Vision Interpretability

2.1 Background

2.1.1 Interpretability Problem

Machine Learning models are used to find abstract patterns within a dataset that would otherwise be difficult to find using traditional programming, thereby approximating solutions to problems that would be nearly impossible to solve otherwise. A major challenge with this method however is the lack of interpretability of ML models. This is because ML models have various parameters that are fine-tuned during training instead of being predefined by a programmer.

2.1.2 Mechanistic Interpretability Overview

Mechanistic Interpretability (MI) is the process of interpreting the decisions and architecture of ML models using algorithmic techniques. It is a rapidly evolving field and one of the "hot topics" in ML as understanding how ML models learn

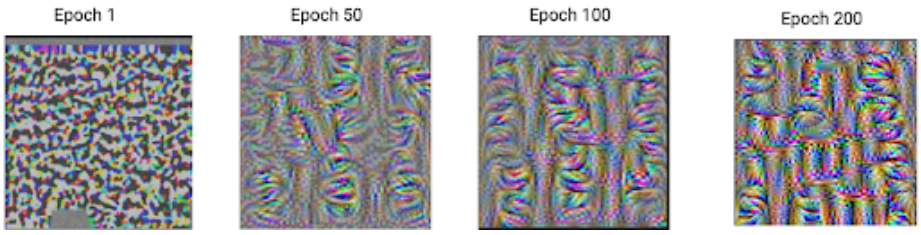


Figure 1: Visualizations by # of Epochs

provides valuable insight into how to train models more effectively and remove biases.

2.1.3 Computer Vision MI

Computer Vision (CV) is generally the field of ML focused on extracting and understanding information from images and video. Because of CV's visual nature, and breakthroughs in the CNN architecture during the early 2010's, the majority of early work in MI was done in CV. A common task in CV is image classification, where a model is given an image and predicts what that image's class is out of a list of classes. An example of an image classification model is a CNN designed to predict whether an image is a dog, cat, or golf cart.

2.2 Methodology

2.2.1 Feature Visualization

A powerful method for interpreting Image Classification models is Feature Visualization. Feature visualization for a unit of a neural network is done by finding the input that maximizes the activation of that unit. A common method for finding this input is "to generate new images, starting from random noise. To obtain meaningful visualizations, there are usually constraints on the image, e.g. that only small changes are allowed. To reduce noise in the feature visualization, you can apply jittering, rotation or scaling to the image before the optimization step" [11]. Both of these techniques were used in generating the images for this paper.

2.2.2 OmniXAI

Code from the OmniXAI GitHub [19], A library for explainable AI, was used to find inputs that maximize features in the VGG16 model, a popular image classification CNN. Figure 1 depicts how OmniXAI's implementation of feature visualization develops an image.

The image starts staticky and very low resolution, and as the number of epochs increases, the resolution of the image increases drastically as the patterns

become clearer. VGG16 has 13 convolutional layers, each with unique maximal activation images for each of its features.

2.3 Results

Below are the results of generating images over 300 epochs for 6 features in each convolutional layer of VGG16. To preserve space, roughly every other convolutional layer has its results included in the visuals below.

There is a general trend in the maximal activation images where the deeper into the network their corresponding channels are, the more complex their images become.

2.3.1 Layer 1

In layer 1 (fig. 7), the images don't seem to be looking for anything in particular other than color. This is likely because at this step in the model, no context has yet been diffused across channels other than the color channels.

2.3.2 Layer 2

In layer 2 (fig. 7), there are some fundamental and very revealing patterns. As shown in the images, the channels in this layer appear to look for vertical, horizontal, or diagonal edges. This suggests that in order to understand the objects in an image, CNNs first find all the edges of objects in the images, which can be crucial for detecting the tmore complex objects in later layers.

2.3.3 Layer 4

In layer 4 (fig. 8), there is still indication that the model is detecting for edges, but these edges seem to be interwoven with each other. For example, in channel 1, it seems to detect diagonal lines going down in the left half and sides of the image, and diagonal lines going up in most of the right half. Channel 4 seems to be detecting for mostly downward angled edges with a particular texture and channel 2 seems to be detecting for some sort of polka dot pattern.

2.3.4 Layer 6

In layer 6 (fig. 8), the textures and edges the channels are detecting for increase in variety, with some complex patterns being detected for in channel 1.

2.3.5 Layer 8

In layer 8 (fig. 9), there is a drastic increase in the complexity of images being detected for, with channels 1 and 4 looking for swirl patterns, channels 0 and 2 looking for cell grid patterns, and channels 3 and 5 looking for a scaly pattern.

2.3.6 Layer 10

In layer 10 (fig. 9), these patterns further melt together. In channel 5 they form strings with gridded texture. In channel 3, they form overlapping squares with a texture of square patches.

2.3.7 Layer 12

In layer 12 (fig. 10), several melted together patterns appear in each image, and their directions are even more varied. There is also an anomaly for channel 3 where a static image is most optimal.

2.3.8 Layer 13

In layer 13 (fig. 10), the most complex images are present in a similar way to layer 12, but with even more varied textures and arrangements.

2.3.9 Layer 12 Anomaly

To explore the anomaly in layer 12 (fig. 10) further, I ran 10 epochs for 64 channels and found that 5 appeared to be fully static. Since it's unlikely this image would remain optimal under translation, I wanted to investigate why this was the case. I ran 100 epochs for 64 channels in layer 13 and found that there were even more fully staticky images, around 20. This leads me to believe that the deeper into the network the layers are, the more epochs it takes to find the maximal activation image. However, channels may also simply be incapable of learning. This appears to be the case for the anomaly in layer 12 as after running 1,000 epochs to generate the image for that channel, it was still pure static.

2.4 Conclusion

Generally, VGG16 appears to detect for simpler features in earlier Conv2d layers and more complex features in later layers. The features it detects for are typically a combination of features from previous layers with various transformations applied and simpler features added. Additionally, each channel appears to detect for distinct patterns, especially in later layers with more abstract features. This suggests the model prevents two channels from forming overlapping feature detection.

3 NLP Interpretability

3.1 Background

3.1.1 Context in NLP

In NLP, a word's context is essential for determining its meaning. For instance, the word "bank" can refer to a river bank when preceded by words referring

to bodies of water, or a financial bank when preceded by words referring to economic institutions. Another example which applies to sentiment analysis is sarcasm. In a sarcastic sentence, subtle structural and semantic clues cause the ground truth sentiment to be very different than the sentiment suggested by the majority of out-of-context words in the sentence.

3.1.2 Integrated Gradient Method

Mechanistic Interpretability has shifted its focus to NLP in recent years, resulting in a variety of new NLP interpretability algorithms being developed. For instance, the Integrated Gradient (IG) algorithm produces a heat map of which words contributed most and least to a model's sentiment classification. It does this by approximating the gradient of the review prediction with respect to a word in the review by comparing the review with the word present versus the review with the word absent. It repeats this in batches for each word in the review to produce a heat map of which words contributed most and least to the review prediction [13].

3.2 Methodology

I created a transformer using the transformer block from the torch.nn library and trained it for 10 epochs to perform sentiment analysis on movie reviews. After each epoch, the integrated gradient method was applied to determine the extent to which words in the review affect the transformer's prediction. Part of the code for training and testing the model was from the OmniXAI library [19]. After each epoch, integrated gradient was used on the transformer and its accuracy was measured using the test dataset. This resulted in periodic data on the evolution of this model's reasoning which I then interpreted.

3.2.1 Dataset

A dataset of 49,582 non-ambiguous movie reviews and their corresponding labels of positive or negative were used [7]. The train/test split was 80/20. Because of the dataset's size and the importance of running integrated gradient to capture the model's reasoning frequently throughout training, each epoch used a random sample of 20% of the training dataset.

3.2.2 Preprocessing

Each word in a given movie review was converted to a unique id. If there was no id corresponding to a word, then it was assigned an id corresponding to unknown words. Any punctuation was removed. Finally, to give each review vector the same number of dimensions, with the id of each word corresponding to a dimension, the end of the review was padded with 0's up to a length of 256. Any reviews longer than 256 words were discarded.

3.2.3 Reviews Used for Interpretability

The model's performance and reasoning was tested after each epoch using 13 reviews of varying sentiment, complexity, and linguistic devices. The reviews can be found in fig. 13.

The numbered order of these reviews is the same as the numbered order of the reviews in figures 16-21. Additionally, these reviews will be referred to using their most prominent feature and ground truth (e.g. simple positive, sarcastic negative, etc.). See fig. 13 to find these names beside their cooresponding review. Each review's number will also appear in parenthesis whenever it is mentioned for ease of lookup in fig. 13 and fig. 16-21.

3.2.4 Architecture/Training Details

Dropout was used to prevent the transformer from overfitting to the data (my testing showed that the model overfitted much quicker when dropout wasn't used). Using GELU, a non-linear layer similar to ReLU but with a smoother change in slope at the origin, also helped prevent the model from overfitting. Weights were initialized using Xavier initialization and biases were initialized to 0 as these initializations enable feed-forward networks to train more efficiently [2].

3.2.5 Output and Prediction Confidence Using Integrated Gradient

While the transformer classifies reviews as positive or negative, it doesn't do so discretely. In reality, it outputs its two predictions for how likely the sentence is positive or negative, which are used by integrated gradient. Integrated gradient then returns all the words in a given review with sentiment scores for each word. Roughly speaking, words with positive scores contribute positive sentiment and words with negative scores contribute negative sentiment according to the model.

Summing all the word sentiment scores in a review will give a rough estimate of the review's sentiment. This method was used to calculate the values in figures 14 and 15. The higher the review's sentiment value, the more positive the model believes a review is. E.g. if the sentiment score is 10 for review A and 50 for review B, then the model predicts that they're both positive, but is more confident that A is positive than B. We will define the model's confidence in a prediction to mean the magnitude of the review's sentiment score.

3.2.6 Understanding Sentiment Prediction by Word Figures

Figures 16-21 were generated by applying integrated gradient. Instance # indicates the # of the review being interpreted and the positive and negative labels following the word "Class" represents the model's prediction for the sentiment of the sentence. It is important to note that the meaning of the green and red word coloring changes depending on whether a review has a positive or negative prediction. Essentially, words highlighted green support the model's prediction

and words highlighted red contradict the model's prediction. E.g. if the model predicts a review is positive, then a word highlighted green helped "convince" the model that the review is positive and a word highlighted red helped "dissuade" the model from its prediction. However, if the model predicts a review is negative, then a word highlighted green helped convince the model that the review is negative and a word highlighted red helped dissuade the model from predicting negative. Finally, the less faded a green or red word appears, the more it had an effect on convincing or dissuading the model. By extension, a gray word effectively has little to no effect on the model's prediction.

3.3 Results

3.3.1 Training Loss and Test Accuracy

The Training Loss (fig. 11) exponentially decayed from 0.45 during epoch 1 to 0.14 during epoch 10. This suggests the model was gradually adjusting its parameters to reach a local minimum, and that a local minimum was reached around epoch 8. The test (fig. 12) accuracy jumps to 81% on the first epoch. This suggests the model was able to learn how to guess correctly on most reviews relatively quickly. Over the next two epochs, test accuracy rose to 88%, implying that the model gradually became better at guessing the sentiment for more challenging movie reviews. It then leveled off at $\tilde{8}8\%$ accuracy, which supports the aforementioned evidence that the model reached a local minimum. The relative drop at epoch 10 seems significant, but it's likely overfitting as the test accuracy remained the same.

3.3.2 Confidence by Classes

The model (fig. 14) in epochs 5-10 consistently predicted three fourths of the positive reviews to be positive with more confidence than any of the other sentences. In a similar manner, while the two mixed reviews fluctuated between positive and negative, the model consistently had low confidence in its predictions for them. The model was far more confident in its negative predictions for negative reviews than any other type of review. However, four out of seven negative reviews were frequently labeled as positive. This suggests that the majority of the model's inaccuracy for these reviews was the result of negative reviews being labeled as positive.

3.3.3 Fluctuations in Confidence

Fig. 13 and 14 show how the sentiment scores of the 13 reviews fluctuated while training. From 0-2 epochs, the sentiment scores for a few of the negative reviews dropped quickly into the negatives. The mixed review that started high fell to 0 where the other mixed review started and stayed at. The positive and a few of the negative reviews that would later become positive stayed around 0.

Despite fig. 12 implying that the model stayed relatively stable from epochs 3-10, fig. 13 and 14 reveal several interesting fluctuations in sentiment scores.

During epochs 5 and 6, the model gained more confidence in its positive and negative predictions. The model's confidence in positive versus negative predictions then strangely diverged during epochs 7-10. During epoch 7, its confidence in reviews with positive predictions fell sharply before rising sharply during epoch 8 and falling sharply again during epoch 9. Meanwhile, its confidence in reviews with negative predictions gradually rose from a minimum around epochs 6-7 to a high of around 0 by epoch 10. This implies it handled positive and negative sentiment detection differently during epochs 7-10, with changes in negative sentiment prediction being gradual and positive sentiment prediction being volatile.

3.3.4 False Positives Overview

Based on fig. 12, the model plateaued at %87-%88 accuracy. To better train the model, we should look for patterns in the reviews it is classifying wrong. The four aforementioned negative reviews which the model often predicts are positive (false positives) are the most consistent source of inaccuracy out of the 13 sentences. These reviews are the subtle (3), sarcastic (4), technical (8), and figurative (10) negative reviews. The negative reviews the model has little trouble classifying are the simple (1), implicit (6), and comparative (7) negative reviews. The sentiment prediction by word for these sentences can be found in fig. 16-21. In the following analysis, fig. 20 will be used. For information about how to interpret these figures, see 2.2.6.

3.3.5 Analyzing Correctly Labeled Negatives

We will start by analyzing how the model was successfully able to classify the simple, implicit, and comparative negative reviews. The simple negative (1) review has only words with negative implications. Because of this, it consistently received the lowest sentiment score with the exception being at epoch 0 (untrained) and epoch 10. Just by memorizing the out-of-context meaning of words, the model is able to easily classify it. The implicit review has less words with negative connotations, but also lacks any words with positive connotations. Because of this, our interpretation is that the model again was able to utilize its learned knowledge of out-of-context word sentiments to consistently classify this as negative. The comparative has a mix of positive and negative words, meaning the model can't rely entirely on out-of-context word sentiments. The words original, remake, and 1984 version which are all involved in comparing a current movie to a prequel have negative connotations in fig. 20. This suggests that the model interprets reviews that bring up a better original in comparison to a remake as negative.

3.3.6 Analyzing False Positives

Now we will discuss the challenges the model faced with correctly classifying the subtle, sarcastic, technical, and figurative reviews. The subtle review (3)

at its core relies on two figures of speech: "never quite finds its footing" and "beautifully wrapped empty box". The model does appear to express some in-context understanding of words in this sentence. For instance, wrapped is ambiguous out-of-context but since it is preceded by beautifully, the model uses the context to recognize wrapped as part of a larger positive phrase, resulting in wrapped being positively interpreted. This similarly occurs with empty box. Box has an ambiguous connotation, but being preceded by empty, the model interprets the phrase and thus box as negative. This suggests the model is able to perform limited in-context learning of nearby pairs of words for both positive and negative sentiment.

Unfortunately, the model never quite finds its footing with correctly interpreting the more complex phrases. The model finds the "beautifully wrapped" and "empty box" phrases, but isn't able to view them as connected into the negative phrase "beautifully wrapped empty box." This similarly occurs with "never quite finds its footing."

The model struggles to correctly interpret sarcasm (4), in particular the word "perfectly" and phrase "brilliant job" are taken literally. While the model correctly interprets other parts of the sentence, these words carry strong positive connotations and therefore result in the model frequently labeling it as a positive review.

The model fails to interpret the technical review because its in-context learning capabilities are limited. Like in the subtle review, it picks up on phrases of length two like "constantly breaks", which is often positive, but not longer phrases like "constantly breaks immersion". This suggests in conjunction with prior analysis that phrases of length two are the longest the model's able to reliably understand.

Finally, the model fails to correctly interpret the very polarized figurative negative review. This review requires very complex context spanning multiple words to understand, which the model still lacks. Therefore, it relies primarily on out-of-context word sentiment, and because there are many words with positive sentiment, it often interprets the review as positive.

3.4 Conclusion

Using the Integrated Gradient technique, we were able to uncover significant amounts of information about our transformer. Much of this information, such as information about what causes negative reviews to become false positives and the extent to which the model can understand phrases of various sizes, can be used to understand where the model should improve and how we could create the right environment for that improvement.

4 Abstract

Recently, Mamba has emerged as a powerful model for sequence prediction. Mamba has been widely studied for its use in both language tasks[4, 18] and

vision tasks[20, 9]. A natural synthesis of these 2 areas is optical character recognition. In addition, Mamba has been found to have in-context learning capabilities comparable to those of transformers while having a much lower overhead[3]. In our experiments, we find that Mamba is able to leverage this in-context learning capability to improve accuracy when labelling single-character images. However, we fail to reproduce these findings for the task of labelling entire words.

5 Background on State Space Models

In this paper, we use a network architecture called Mamba, which is a variant around a family of sequence-to-sequence models called a state space models(SSMs).

5.1 State Space Models

State space models(abbreviated SSMs) are a type of sequence-to-sequence model that have recently found promising applications in language modeling[4], image processing[20], and more[5].

There are two broad categories of SSMs: continuous SSMs and discrete SSMs. Discrete SSMs based on continuous SSMs, so it is important to first understand continuous SSMs. Continuous SSMs were introduced by Kalman[12] as a generalization of certain signal filters. Given an input signal x, which is a vector that depends on time, it produces an output signal y, which is another(possibly differently-sized) vector that depends on time based on the following formula:

$$\frac{ds}{dt} = \mathbf{A}\vec{s} + \mathbf{B}x$$
$$y = \mathbf{C}\vec{s} + \mathbf{D}x$$

, where $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ determine the specific dynamics of the system. One interpretation is that continuous SSMs model an internal state s that is linearly affected by the input signal x, and produce an output signal y based on x and s. Another perspective is that continuous SSMs are a generalization of matrix differential equations, where the system is driven by an input signal x and the output signal is projected, possibly destroying information.

These models can simulate a variety of systems, such as physics equations, financial equations, and more.

5.1.1 Discretization

Since we can't store general continuous sequences in hardware, real implementations of SSMs approximate the dynamics across discrete timesteps. A common

method for discretizing SSMs was introduced by Gu et al. [5]

$$s_k = \overline{\mathbf{A}} s_{k-1} + \overline{\mathbf{B}} x_k \quad \overline{\mathbf{A}} = \left(I - \frac{\Delta}{2}\mathbf{A}\right)^{-1}\left(I + \frac{\Delta}{2}\mathbf{A}\right)$$
$$y_k = \overline{\mathbf{C}} s_k \qquad \overline{\mathbf{B}} = \left(I - \frac{\Delta}{2}\mathbf{A}\right)^{-1}\Delta\mathbf{B} \qquad \overline{\mathbf{C}} = \mathbf{C}$$

, where Δ is the timestep. This scheme assumes that the input signal is constant within each timestep and uses the trapezoidal rule for integration. Note that $\mathbf{D}$ is omitted since many model architectures, such as Mamba, include skip connections, which make $\mathbf{D}$ redundant.

5.2 Hyperparameters

For the specific SSM architectures that we will discuss, the core SSM layers will all be setup so they split the input signal dimension-wise into many 1-dimensional signals and feed them individually through parallel SSMs. This means that all of the SSMs will have an input and output dimension size of 1. This is done to maximize the internal state size.

In the style of Gu et al. [5], we use the following notation for the remaining hyperparameters:

- B - The batch size.
- L - The length of the sequence.
- D - The number of individual SSMs to run in parallel. This is the actual number of input channels.
- N - The state size for each SSM.

5.3 S4

S4[5] is one implementation of state space models that implements the scheme detailed above. S4 uses an initialization for $\mathbf{A}$ based on the HiPPO framework, which is designed to efficiently represent long-range dependencies. The key development introduced by S4 is that the S4 parametrizes $\mathbf{A}$ as a normal-plus-low-rank(NPLR) matrix – a matrix that can be written as

$$\mathbf{A} = \Lambda - \mathbf{D}\mathbf{D}^*$$

, where Λ is a diagonal matrix and $\mathbf{D}$ is a column vector. The authors then use the spectral properties of these matrices in an algorithm that brings the training complexity from $O(BLN^2)$ down to $O(BN(N\log N + L\log L) + B(L\log L)N)$[5]. This allows the model to use 99.7% less memory compared to LSSLs, another popular SSM model at the time. Using the increased model size made feasible with the efficient algorithm, they were able to achieve SOTA on the task LRA Path-X [14].

5.4 Mamba

Mamba[4] is the model architecture that we use in our experiments. Mamba introduces a new SSM architecture, called S6, and introduces a new block design around this SSM layer for increased selectivity.

The S6 architecture is an improvement on the S4 architecture that introduces timesteps that vary based on the input data. This breaks the efficient algorithm introduced in S4, so the authors design a hardware-aware algorithm to maintain performance with large state sizes.

The authors also introduce a new block model, which improves on the existing H3[1] architecture and adds explicit selection functionality.

The main advantage enabled by these changes is an increased ability to selectively ignore data. The authors propose several mechanisms for these effects, but the paper focuses on two primary mechanisms. Firstly, variable timesteps allow the model to selectively "forget" and "ignore" specific sequence elements. If the timestep for a given input is set to a high value, the existing state will decay to close to 0, allowing the model to "forget" all previous information. If the timestep for a given input is set to a low value, the state has very little time to be influenced, and in the case of the timestep being 0 or almost 0, the model effectively skips the given input token. Secondly, the explicit gating path in the block architecture allows the model to set specific output tokens to 0.

The authors find that this selectivity allows the model to generalize well on long-range copying tasks, as the model learns to skip over irrelevant tokens. It has also been found that Mamba is is capable of in-context learning for natural language tasks[3], which is we hypothesize generalizes to optical character recognition tasks.

6 Methodology

In this section, we explain our setup for determining Mamba's ability to learn from context. In our experiments, we test 2 different models which attempt to use Mamba for its in-context learning abilities against a variety of datasets, elaborated on below.

6.1 Data

In this paper, we train our models to predict text labels in a type of task that we will call **in-context OCR**. A single instance of an in-context OCR task consists of a set of images of text, and the task is to assign labels to these images. The difference between in-context OCR and standard OCR is that with in-context OCR, each instance consists of multiple images in the same context. This means that in theory, an agent predicting this task can benefit by using context clues from different parts of the image.

With our models, we train them to solve this task in an autoregressive manner. That is, the prediction for each character in each label is based on the previous characters and previous image/label pairs.

In all of our datasets, we rescale each image to a height of 32 pixels.

6.1.1 Position Encoding

An issue with in-context OCR as outlined above is that the model is not provided any information about the scale and relative position of each word. As such, we append rotational positional encodings similar to the ones used in the transformer architecture[16] in each word such that each pixel embeds its relative position within the original image. With these positional encodings, a model can now directly access the position of any given image, and it can infer the scale of each image by evaluating the change in position between the corners of the image.

6.1.2 Text Injection

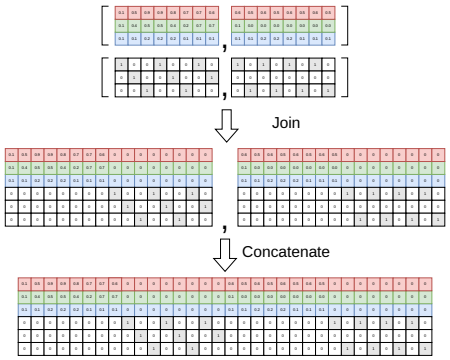


Figure 2: In-Context Encoding

In order to supply context to the models, we encode our dataset using a scheme that involves what we will call in-context encoding. Before in-context encoding, all of the images within a context need to be transformed into sequences. We will call these sequences the image matrices, where each column corresponds to a pixel in the image(with positional encodings and other per-pixel metadata) Then, we encode the labels as one-hot tokens such that each label is represented as a matrix of one-hot columns. We now apply the in-context encoding transformation on these 2 lists of tensors. First, we diagonally append the label tensors to the image sequence tensors such that they are in separate channel spaces. Then, we append all of the combined tensors lengthwise to get the final tensor. This process is shown in Figure 2.

With this encoding scheme, we assume that the data is being processed by an auto-regressive model, where each output token is evaluated based on the previous (correct) output tokens, and previous predictions in context.

We hypothesize that Mamba's ability to do in-context learning[4] will allow model architectures containing Mamba to perform well on in-context image prediction.

6.1.3 MSCOCO-Text

The first dataset that we use is the COCO-Text dataset, based on the MS COCO dataset[17][8]. This dataset consists of a set of 63k images with each instance of text labeled with the text string, an axis-aligned bounding box, and a bounding polygon.

The processing that we perform in order to convert this into an in-context OCR task is fairly straightforward. First, we filter the dataset to only contain "normal" examples. In particular, we remove instances that have any of the following problems:

- Instance resolution is too low(AABB height less than 32).
- Aspect ratio is bad(width is not between 0.5 and 1.5 times the height times the number of characters).
- Label is a single character or contains more than 10 characters(since single characters are usually hard-to-discern punctuation and more than 10 characters could stretch the models' ability to memorize).

Next, we remove any contexts that contain 1 or 0 words. This is since these instances don't effectively test in-context learning. Unfortunately, this still misses a few bad samples and shrinks the dataset down to 439 contexts, which greatly increases the risks of overfitting. After filtering, for each text instance, we crop the image to the AABB corresponding to the instance. If the bounding polygon happens to be a quadrilateral, we instead project the bounding polygon onto a rectangle with the same dimensions as the AABB. Then, we proportionally scale the cropped image so that the height is 32. Finally, for each pixel, we calculate the coordinates that we want to encode using to the following formulas:

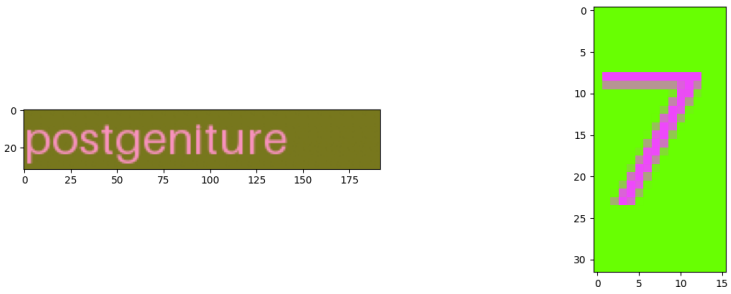
$$x_{\text{relative}} = x \qquad x_{\text{absolute}} = \frac{x_{BB} + x \cdot w_{BB}}{w}$$

$$y_{\text{relative}} = y \qquad y_{\text{absolute}} = \frac{y_{BB} + y \cdot h_{BB}}{h}$$

, where x and y are the normalized coordinates of the pixel within the cropped images($x \in [0,1]$, $y \in [0,1]$), w, h are the width and height of the image in pixels, and $x_{BB}, y_{BB}, w_{BB}, h_{BB}$ are the left edge, top edge, width, and height of the AABB in pixels.

We append these positional encodings as new channels alongside the red, green, and blue channels.

For the labels, we encode each character as a one-hot token and add a special ending character so that the model learns to terminate words.



(a) An example image from the synthetic words dataset

(b) An example image from the synthetic digits dataset

Figure 3: Examples from the synthetic datasets

6.1.4 Synthetic Words

Since the COCO-Text dataset contains very few "good" images, we construct a synthetic dataset that serves as a toy example for in-context tasks. In this dataset, each context is generated using 10 words randomly sampled from the GNU Collaborative International Dictionary of English[15] along with a random foreground and background color. Each image consists of the text printed on a solid background, with the aforementioned colors. Images within the same context have the same foreground and background color, but images in different contexts might have different colors. Like with COCO-Text, the label strings in this datasets are ended with a special terminator character. An example is shown in Figure 3a. It is also important to note that positional encodings are omitted since there is no "original" image that the text instances are placed in.

In-context learning for this dataset would mean memorizing what foreground/background mean within a dataset.

6.1.5 Synthetic Digits

This dataset is similar to the synthetic words dataset, except that instead of entire words, this dataset is just images of single letters. An example is shown in Figure 3b

6.2 Validation Set

In each of our tests, we validate the models against 2 different datasets: the in-context validation set, and the shuffled-context validation set. The in-context validation set is just a smaller set of contexts sampled from the same population as the training data. The shuffled context validation set is a modified version of the in-context validation set where the instances are shuffled with the following algorithm;

1. Initialize the output set as a copy of the in-context validation set.
2. Let the current instance be the first image/word pair in the first context in the output set.
3. Pick a random image/word pair from the in-context validation set.
4. Set the current instance to be the chosen image/word pair.
5. Repeat steps 2-4 for all instances image/word pairs in the output set.

The resulting set contains contexts that have the same shape as the original contexts, but the actual image/word pairs in the shape are unrelated to each other. This validation set tests whether the model is using the context for text prediction. Models that ignore the previous image/label pairs from the context are expected to perform the same on both sets, since individual image/word pairs are left unchanged by this shuffling procedure. Models that rely on previous image/label pairs are expected to perform worse, since at best, they lose the ability to learn from context, and at worst, they actually learn incorrect information from the unrelated images.

6.3 Models

In our experiments, we evaluate 2 models.

6.3.1 Multi-Layer Mamba

The first model architecture that we test is a multi-layer Mamba architecture. The method used for converting images is column-wise flattening. That is, we map pixels in the image to tokens in the sequence such that each pixel is placed next to the pixel directly above and below. The full architecture is shown in Figure 4.

6.3.2 MedMamba

The second model architecture we look at is based on the MedMamba architecture[20]. The main difference between MedMamba's architecture and our architecture is that We modify the SS2D block(boxed in red in Figure 5a) to include context injection for each scanning path. Since context injection adds new channels, we also add a linear layer to the output of the SS2D block to ensure that the number of channels matches the number of channels in the gating vector.

Like MedMamba, we use Patch Merging layers(originally introduced by the SWin Transformer architecture [10]).

6.4 Hyperparameters

For multi-layer Mamba, we set the `d_model` and `d_state` both to 256. We set the layer count to 4 and set the intermediate size to 128.

For the MedMamba stack, we set the input dimension to 16, we set `d_state` to 256 for all mamba layers, and set `d_model` to 256 for the final Mamba layer. We also set the drop path probability to 0.5.

7 Results

As shown in Figure 6a and Figure 6b, both the sequence stack model and the medmamba-based model perform better when validated on in-context data as opposed to shuffled contexts.

However, as shown in Figure 6c and Figure 6d, the in-context accuracy boost disappears. In addition, we find that the overall performance degrades significantly. For MedMamba, the accuracy dropped from 100% down to around 98%. For sequence stack, the accuracy dropped from 100% down to around 40% for in-context, and 80% to 40% for shuffled-contexts.

With the COCO-Text datasets, this trend worsens, likely due to the small dataset size. In particular, MedMamba now fails to achieve high accuracy in either validation set.

One possible explanation is that the multi-character task requires Mamba to store far more information in its internal state. With single letters, the model can theoretically do the task by just remembering the image data for the previous character. However, with entire words, the model might have to memorize over 10 times the amount of data. In addition, the amount of data that the model is required to memorize varies from word-to-word.

Previous work [6] has found the Mamba models generalize well when tasked with copying fixed-length subsequences over varying distances, but they fail to generalize when the length of the subsequence is varied. An interpretation of these finding is that a Mamba and other SSMs are severely limited by memory constraints. In the single digit dataset, the model only strictly needs to memorize the the image corresponding to a single digit. However, with the word dataset, the model might have to memorize over ten times the amount of information compared to in the single digit dataset. This is since the model has to store all of it's letter labels until the end of the word, when it can output its prediction and forget predictions. This means that with the single digit case, the model can easily memorize the current image with spare memory to store context info. However, in the whole word dataset, the model might simply be training to memorize larger and larger words, leaving no free memory for memorizing context-specific details.

We also found that the MedMamda-based model does significantly better compared to the sequence stack model. This could be due to the inclusion of the CNN layers and down-sampling, which could allow the model to more efficiently extract different features in the image.

The same analysis applies to the results for COCO-Text.

8 Conclusion

In our paper, we demonstrate that the Mamba architecture is capable of leveraging its in-context learning abilities to improve its accuracy on synthetic single-character OCR. We also observe a breakdown of this in-context accuracy boost(and a breakdown of accuracy in general) with both real and synthetic full-word OCR datasets.

This also leaves a few questions that could be the subject of future research. How does MedMamba perform on real-life datasets similar to the synthetic digits dataset(i.e. individual labeled characters in context)? Will in-context learning return with larger model sizes? Is in-context learning capable of beating SOTA OCR models?

References

[1] Daniel Y. Fu et al. *Hungry Hungry Hippos: Towards Language Modeling with State Space Models*. 2023. arXiv: 2212.14052 [cs.LG]. URL: https://arxiv.org/abs/2212.14052.

[2] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, 2010, pp. 249–256. URL: https://proceedings.mlr.press/v9/glorot10a.html.

[3] Riccardo Grazzi et al. *Is Mamba Capable of In-Context Learning?* 2024. arXiv: 2402.03170 [cs.LG]. URL: https://arxiv.org/abs/2402.03170.

[4] Albert Gu and Tri Dao. "Mamba: Linear-Time Sequence Modeling with Selective State Spaces". In: *arXiv preprint arXiv:2312.00752* (2023).

[5] Albert Gu, Karan Goel, and Christopher Ré. "Efficiently Modeling Long Sequences with Structured State Spaces". In: *The International Conference on Learning Representations (ICLR)*. 2022.

[6] Samy Jelassi et al. *Repeat After Me: Transformers are Better than State Space Models at Copying*. 2024. arXiv: 2402.01032 [cs.LG]. URL: https://arxiv.org/abs/2402.01032.

[7] N. Lakshmipathi. *IMDB Dataset of 50K Movie Reviews*. Kaggle Datasets. Version 1. Original data source: Maas, A.L., et al. from Stanford University. 2017. URL: https://www.kaggle.com/datasets/lakshmi/imdb-dataset-of-50k-movie-reviews.

[8] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV]. URL: https://arxiv.org/abs/1405.0312.

[9] Yue Liu et al. *VMamba: Visual State Space Model*. 2024. arXiv: 2401.10166 [cs.CV]. URL: https://arxiv.org/abs/2401.10166.

[10] Ze Liu et al. *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*. 2021. arXiv: 2103.14030 [cs.CV]. URL: https://arxiv.org/abs/2103.14030.

[11] Christoph Molnar. *Interpretable Machine Learning*. 2024. URL: https://christophm.github.io/interpretable-ml-book/cnn-features.html.

[12] Kalman RE. "A new approach to linear filtering and prediction problems". In: *Journal of Basic Engineering* (1960).

[13] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. *Axiomatic Attribution for Deep Networks*. 2017. arXiv: 1703.01365 [cs.LG]. URL: https://arxiv.org/abs/1703.01365.

[14] Yi Tay et al. *Long Range Arena: A Benchmark for Efficient Transformers*. 2020. arXiv: 2011.04006 [cs.LG]. URL: https://arxiv.org/abs/2011.04006.

[15] GNU Dico Team. *GNU Collaborative International Dictionary of English*. 2008. URL: https://gcide.gnu.org.ua/about.

[16] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[17] Andreas Veit et al. *COCO-Text: Dataset and Benchmark for Text Detection and Recognition in Natural Images*. 2016. arXiv: 1601.07140 [cs.CV]. URL: https://arxiv.org/abs/1601.07140.

[18] Roger Waleffe et al. *An Empirical Study of Mamba-based Language Models*. 2024. arXiv: 2406.07887 [cs.LG]. URL: https://arxiv.org/abs/2406.07887.

[19] Wenzhuo Yang et al. "OmniXAI: A Library for Explainable AI". In: (2022). DOI: 10.48550/ARXIV.2206.01612. arXiv: 206.01612. URL: https://arxiv.org/abs/2206.01612.

[20] Yubiao Yue and Zhenzhang Li. *MedMamba: Vision Mamba for Medical Image Classification*. 2024. arXiv: 2403.03849 [eess.IV]. URL: https://arxiv.org/abs/2403.03849.

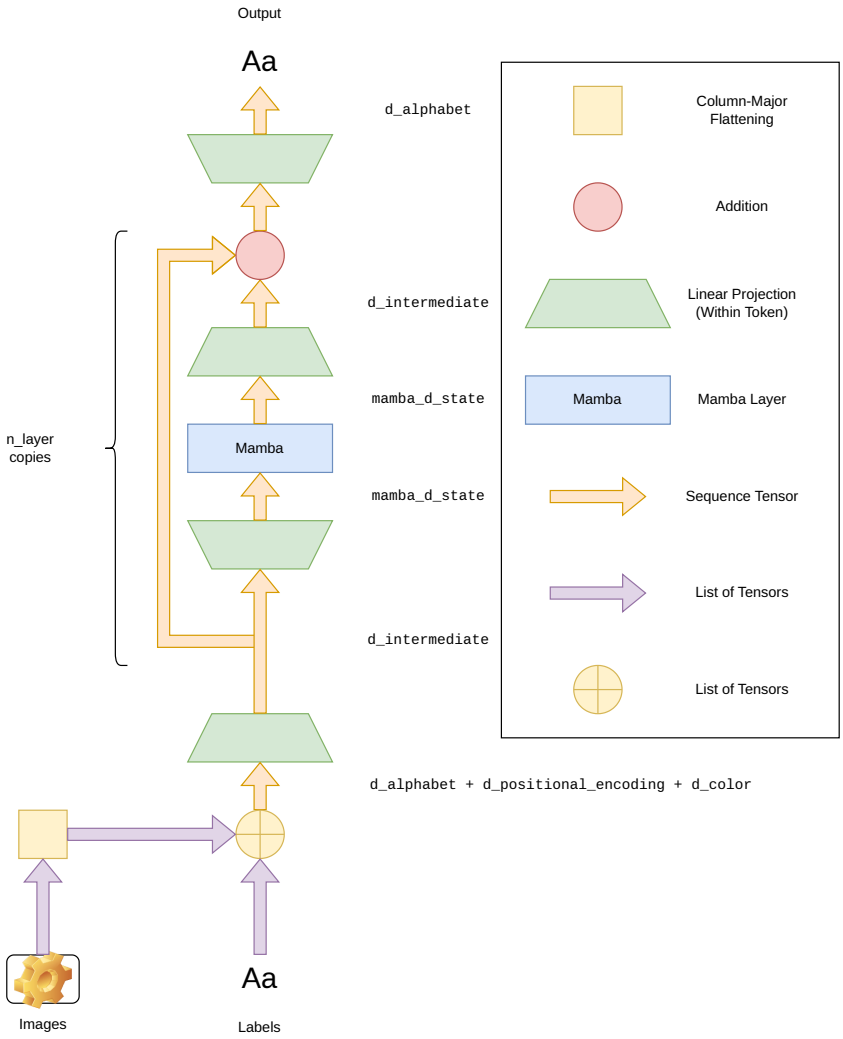
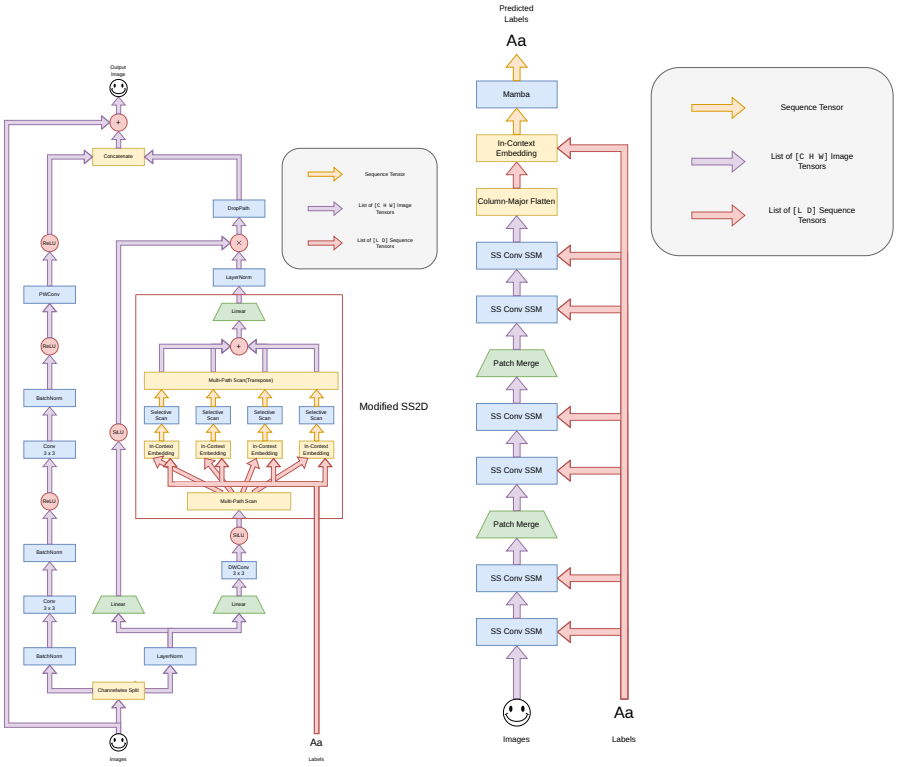


Figure 4: Multi-Layer Mamba Architecture



(a) SS-Conv-SSM with Context Injection (b) High-Level MedMamba stack structure

Figure 5: MedMamba

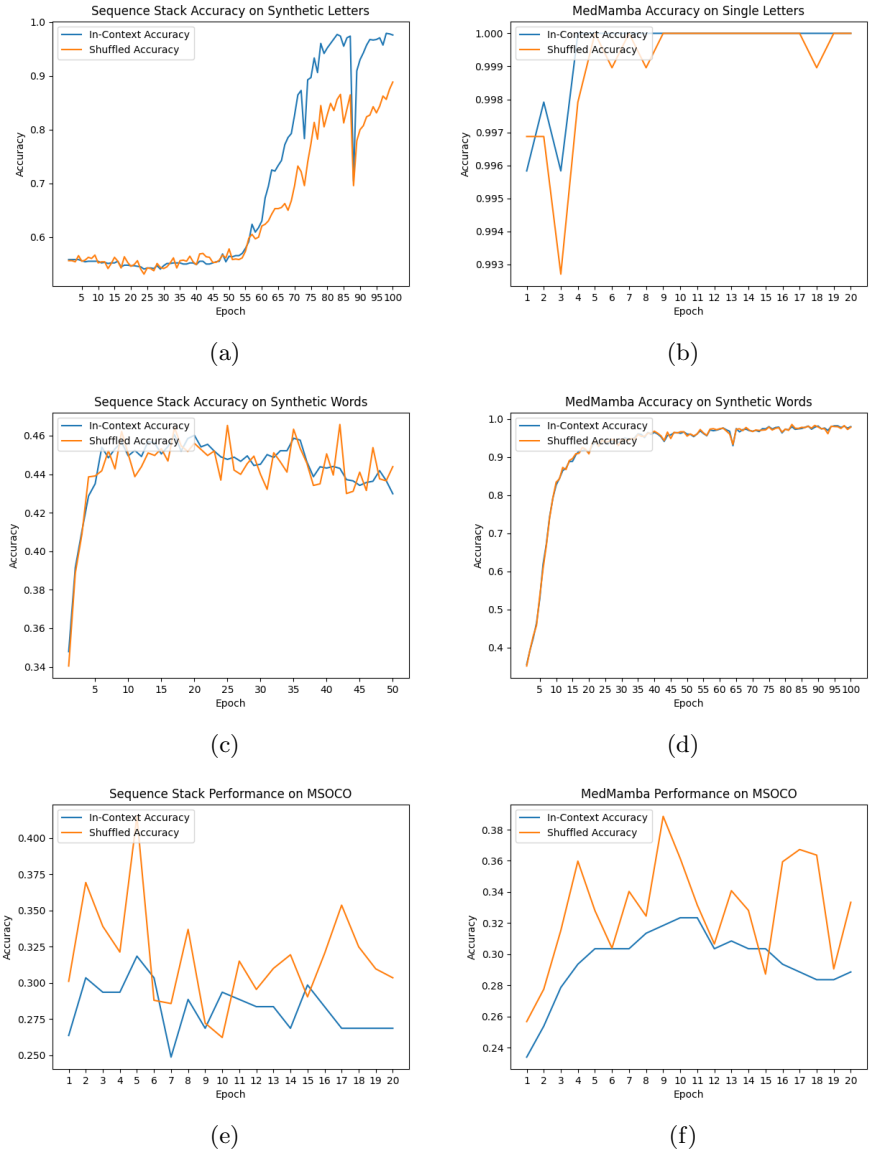


Figure 6: Graphs of results

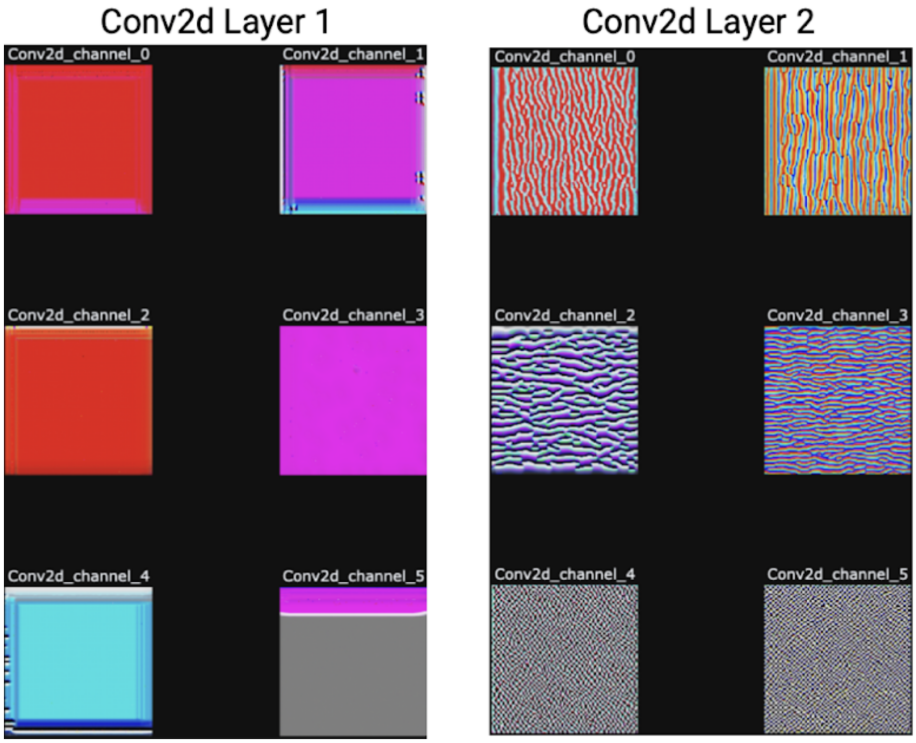


Figure 7: Feature Visualization: Layers 1 & 2

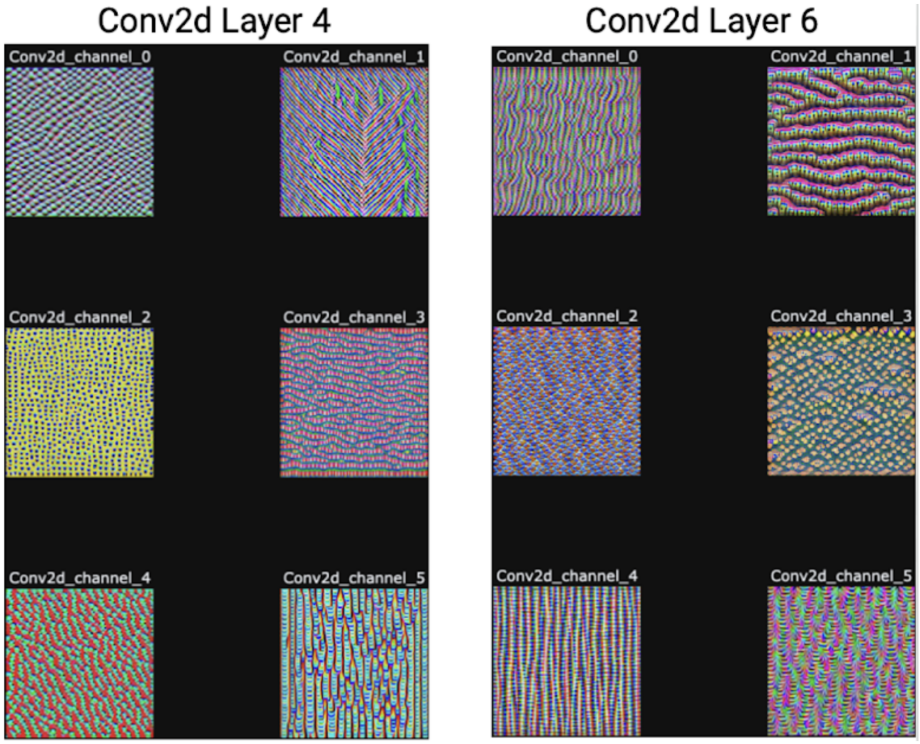


Figure 8: Feature Visualization: Layers 4 & 6

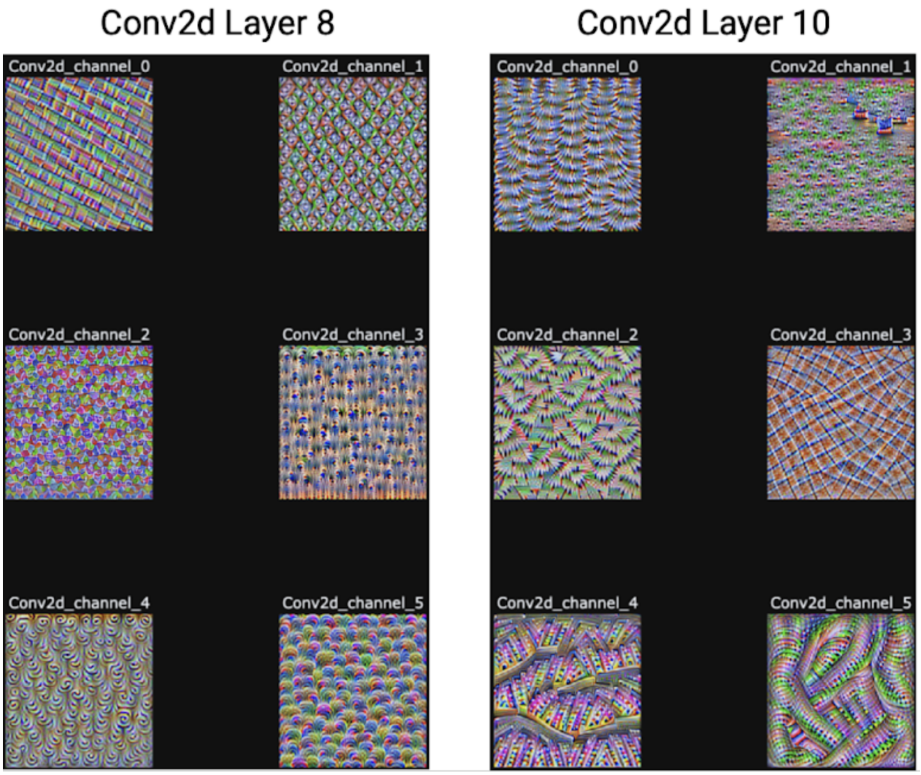


Figure 9: Feature Visualization: Layers 8 & 10

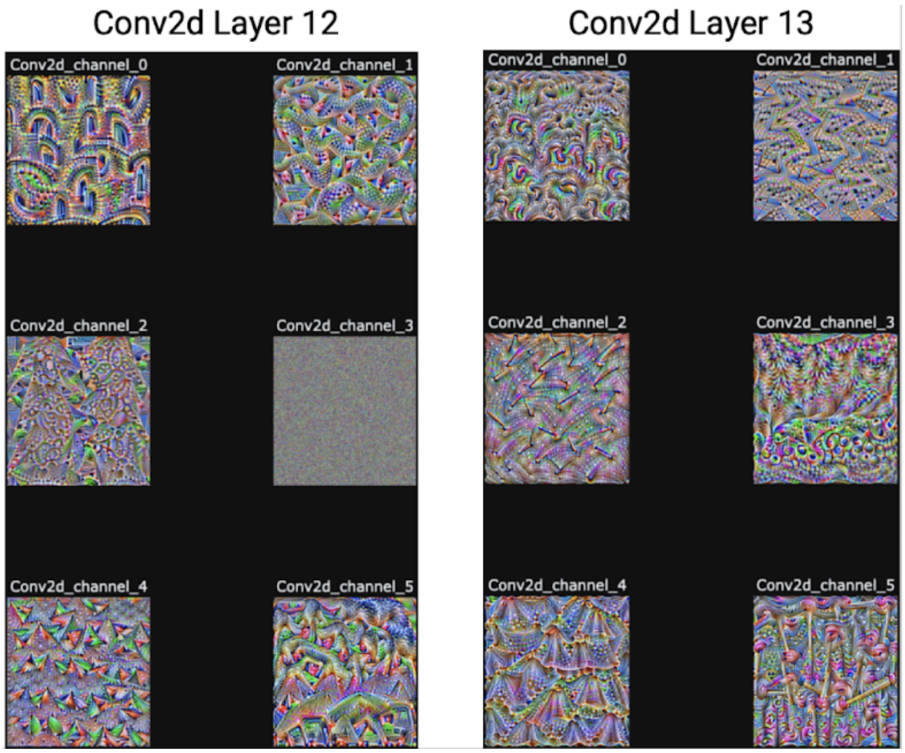


Figure 10: Feature Visualization: Layers 12 & 13

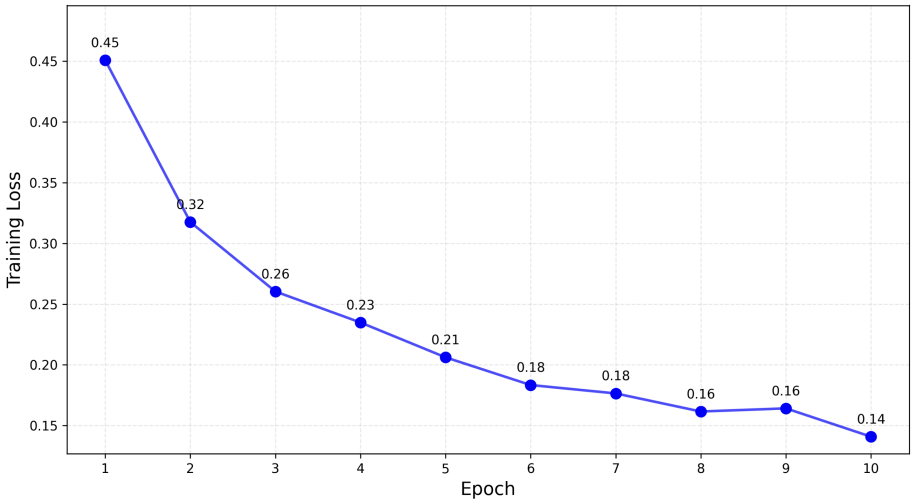


Figure 11: NLP MI Training Loss

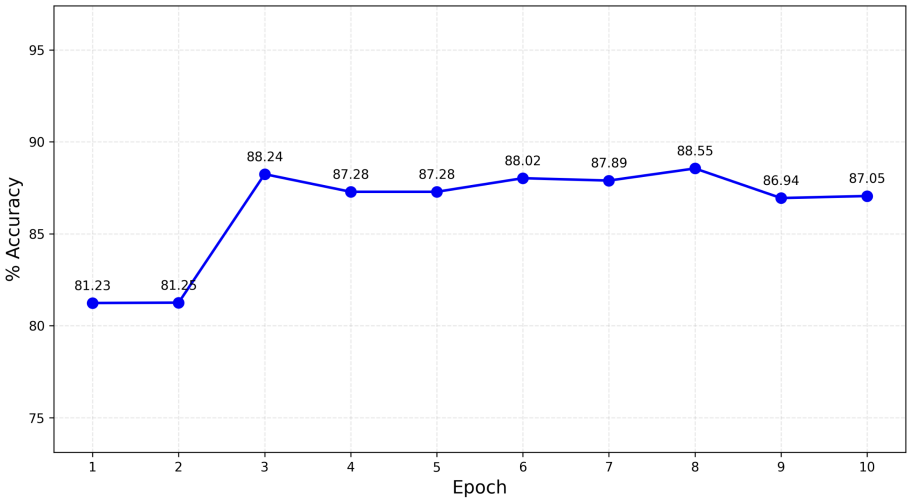


Figure 12: NLP MI Test Accuracy

0. Simple positive: "This movie was great! I loved every minute of it. The acting was fantastic and the story kept me engaged throughout."
1. Simple negative: "This movie was terrible. I hated it. The acting was awful and the story made no sense."
2. Subtle positive: "While not breaking any new ground, this film manages to deliver its familiar story with enough charm and sincerity to make it worthwhile. The characters feel authentic, even when treading well-worn paths."
3. Subtle negative: "For all its technical sophistication and talented cast, the film never quite finds its footing. What could have been a compelling narrative ends up feeling like a beautifully wrapped empty box."
4. Sarcastic negative: "Oh yeah, this is exactly what cinema needed - another mindless sequel that perfectly demonstrates how to take everything that made the original special and turn it into a soulless cash grab. Brilliant job!"
5. Polarized ambiguous: "The first half of the movie is a masterclass in tension building, with performances that will likely earn Oscar nominations. Unfortunately, the final act completely falls apart, abandoning all the careful groundwork for a series of increasingly ridiculous plot twists that undermine everything that came before."
6. Implicit negative: "The director clearly watched a few Kubrick films before making this. Every shot screams 'artistic vision' so loudly you can barely hear the dialogue. The cinematographer must have had a protractor permanently attached to their camera."
7. Comparative negative "Unlike the original, which balanced humor with genuine emotional depth, this remake strips away all nuance in favor of cheap laughs. Where the 1984 version took time to develop its characters, this one seems afraid to let a minute pass without a punchline."
8. Technical negative: "The 48fps presentation creates an uncanny valley effect that constantly breaks immersion. Combined with the aggressive color grading and rapid editing, the technical choices actively work against the period setting's authenticity."
9. Genre-aware positive: "For a low-budget indie shot in 12 days, this is remarkable. Yes, you can see the seams sometimes, and some performances are rough around the edges, but considering the constraints, what they've achieved is impressive."
10. Figurative negative: "I simultaneously admire this film's ambition and resent its pretension. Every frame is composed with meticulous attention to detail, yet somehow it all feels hollow. It's technically perfect and emotionally vacant - a puzzle that's more interesting to discuss than to actually watch."
11. Interpretational mixed: "As a horror movie, it fails spectacularly - the scares are predictable, the atmosphere is non-existent. But viewed as a dark comedy, it's accidentally brilliant. I'm not sure if I should be criticizing its failure or praising its unintentional success."
12. Temporal positive: "Initial viewing left me cold and frustrated. Second viewing revealed layers I'd missed. Third viewing confirmed this as one of the most thoughtfully constructed films of the decade. Like the best art, it demands and rewards patient engagement."

Figure 13: Reviews Used for Interpretability

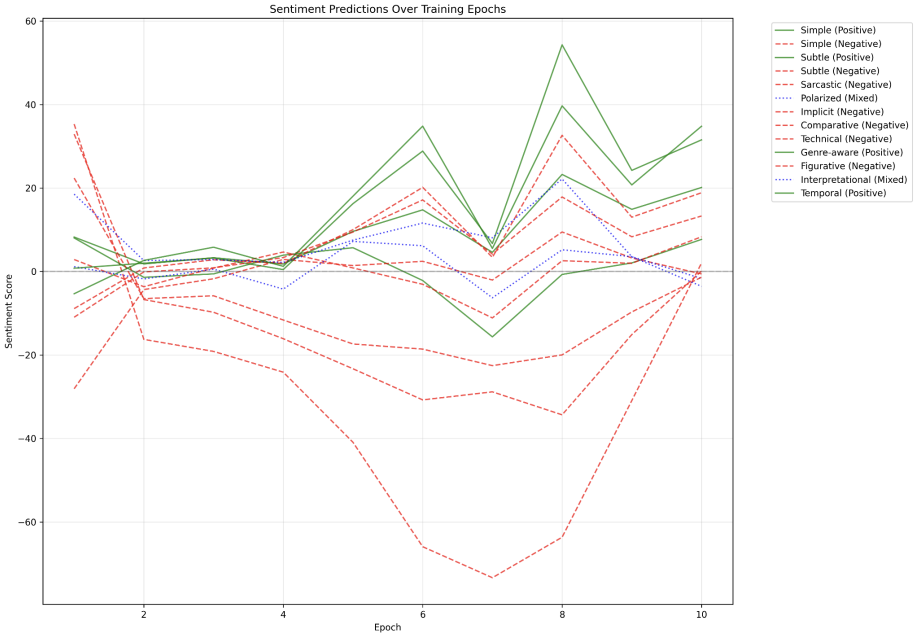


Figure 14: Predicted Sentiment of Sentence Classes by Epoch: Positive vs. Negative vs. Mixed

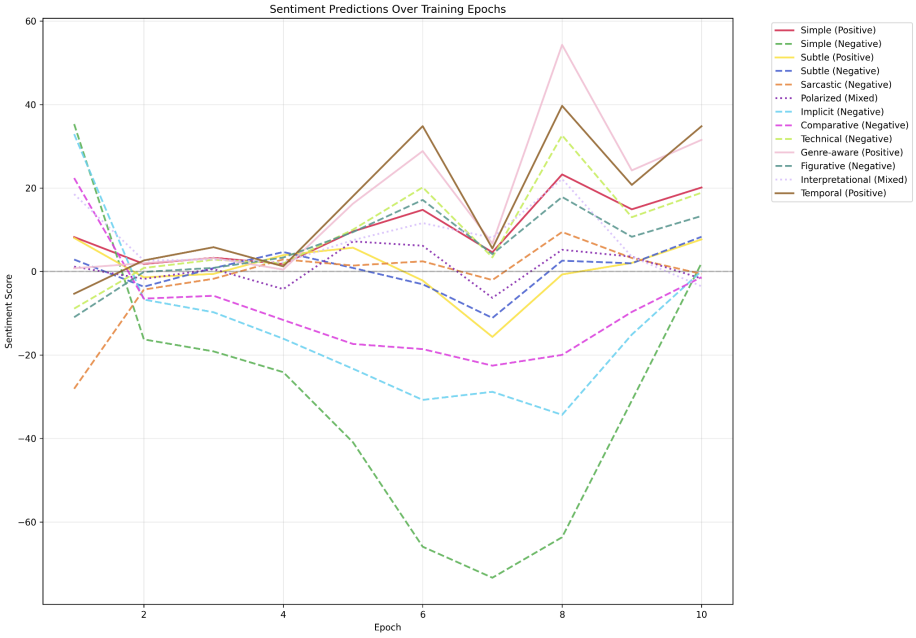


Figure 15: Predicted Sentiment of Sentences by Epoch

Figure 16: Sentiment Prediction by Word: Untrained

Figure 17: Sentiment Prediction by Word: Epoch 1

Figure 18: Sentiment Prediction by Word: Epoch 2

Figure 19: Sentiment Prediction by Word: Epoch 3

Figure 20: Sentiment Prediction by Word: Epoch 8

Figure 21: Sentiment Prediction by Word: Epoch 10