Understanding Vision Transformers In Sparse Data Settings

Aayush Agrawal
Stanford University
aayush2k@stanford.edu

Elliot Dauber
Stanford University
dauber23@stanford.edu

Finn Dayton
Stanford University
finndayton@stanford.edu

Abstract

The recent success of the Vision Transformer (ViT) has made it a popular choice for various computer vision tasks, even as a replacement. However, understanding the inner workings of this complex model can be challenging. In this paper, we propose a novel approach for visualizing the saliency maps generated by the ViT for image classification, as well as an approach for quantitatively measuring the quality of the saliency maps of the ViT. Our method provides an intuitive and interpretable way to analyze the learned representations and attention mechanisms of the ViT. We demonstrate the effectiveness of our approach by altering a custom implementation of ViT in multiple ways and show how this can reveal interesting insights about the model's behavior. In addition, we show how our visualizations can be used to diagnose and identify potential issues with the model, such as overfitting and underfitting. To stress test our approach, we operate in a data sparse environment. Our approach provides a valuable tool for researchers and practitioners to gain a better understanding of the ViT and to improve its performance on various computer vision tasks.

1. Introduction

The recent success of the Vision Transformer (ViT) [5] has propelled it to a popular choice for a wide range of computer vision tasks. Its ability to compete with, and even replace, Convolutional Neural Networks (CNNs) has generated significant interest in the computer vision community. However, comprehending the inner workings of these complex models and understanding the mechanisms that enable their impressive performance remains a challenge.

In this paper, we present a novel approach for visualizing the saliency maps of the ViT architecture, as well as its variants. By leveraging these visualizations, we aim to shed light on the intricate interactions within the model and gain insights into how it processes and attends to different image regions. Through our visualizations, we seek to identify patterns and correlations that can elucidate the decision-making process of the ViT, providing researchers and practitioners with a valuable tool for model analysis and interpretation. The input to our algorithm will be a series of images (animals), and we will use a ViT to output a predicted label. The visualizations (or saliency maps) will highlight areas of the image that the ViT ”concentrated” on during the inference process.

We operate in an intentionally data-sparse environment. Saturating a model with tens of thousands or even hundreds of thousands of images, can lead even poor architectures to decent performance [9]. By constricting the number of training examples from each class–a situation many researchers find themselves in–we hope to better contrast the robustness of different model architectures. Furthermore, we present a quantitative approach to analyzing the saliency maps of vision transformers that aids in our overall goal of understanding vision transformers at a deeper level. This method takes advantage of the powerful image instance segmentation model Fast R-CNN [8] to create masks for the important objects in an image and compares the saliency maps processed by our visualization software with the segmentation masks in the image. While not perfect, this method seeks to understand whether or not a vision transformer is ”paying attention” to the objects we expect it to.

By applying our visualization approach to a benchmark dataset, we demonstrate the effectiveness of our method in revealing intricate details of the models' internal workings. Additionally, we showcase how our visualizations can be utilized - in combination with a host of standard quantitative metrics such as accuracy, precision, recall, and AUC-ROC, to diagnose potential issues with the models, such as overfitting or underfitting, and propose potential improvements.

Overall, our research aims to contribute to the computer vision community by providing a comprehensive visualization method for understanding how vision transformers work. Through our visualizations, we hope to uncover valuable insights into the models' decision-making processes, enhance their performance, and inspire further advancements in the field of computer vision.

2. Related Work

The three papers selected for this literature review provide valuable insights into the state-of-the-art in computer vision transformers, specifically Vision Transformers (ViT) and Convolutional Vision Transformers (CvT), which are the basis of the models we will be visualizing.

The first paper, ”An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale” [5], introduces the ViT architecture as a powerful alternative to Convolutional Neural Networks (CNNs) for image recognition tasks. The ViT replaces the convolutional layers of CNNs with self-attention mechanisms, allowing the model to attend to different regions of the image simultaneously. Specifically, the authors apply the self-attention mechanisms that have become popular in NLP to images by splitting the images into patches, where each patch attends to every other patch in an image. The authors show that ViT achieves state-of-the-art performance on several image recognition benchmarks, including ImageNet, and demonstrate its scalability by training it on large datasets with up to 300 million images.

The second paper, ”CvT: Introducing Convolutions to Vision Transformers” [12], introduces the CvT architecture, which combines the strengths of CNNs and Transformers. The authors argue that the combination of convolutional layers and self-attention mechanisms leads to better performance and scalability than either architecture alone. One of the important insights from this paper is that ViT doesn’t do as well as traditional CNNs when trained on smaller datasets, and that adding convolutional aspects to ViT can improve its performance on small datasets because of the inherent image understanding capabilities that convolutions bring to the architecture. Though, we do not add convolutions to our model, this paper highlighted the need to train ViTs on larger datasets relative to CNNs.

The third paper, ”Do Vision Transformers See Like Convolutional Neural Networks?” [11], investigates whether ViTs and CNNs attend to similar image regions when making predictions and introduces some methods for visualizing transformer attention patterns that we will draw from. The authors find that both architectures attend to similar regions of the image and the differences in performance between them can be attributed to the differences in their learned representations. This will give us a starting point in determining how to visualize the learned weights of the models that we analyze.

3. Methods

Our overall method involved manipulating various elements of vision transformers and analyzing the changes to the visualizations. Before any additional description of the method, we must first thoroughly understand the ViT – as it is a novel architecture. We show a diagram of the model in Figure 1. We used Python for implementation, and used Pytorch for running and training the model [1], Scikit-Learn for analyzing the metrics [2], and Matplotlib for generating the visualizations and graphs [3].

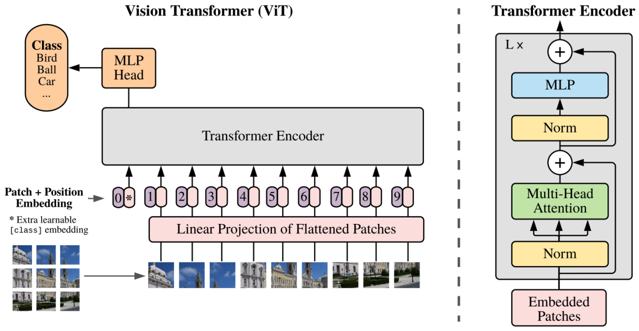


Figure 1: Vision Transformer Architecture [5]

3.1. Transformer Primer

ViTs partition an image into a sequence of non-overlapping, equally-sized patches, subsequently transformed into one-dimensional patch embeddings through a linear transformation. Whereas convolutions in CNNs inherently encode the spatial relationships of the input image, the transformer encoder does not do this. Therefore, to ensure the preservation of spatial relationships within an image, ViTs add positional embeddings to the patch embeddings. This injects information about relative positions of the image patches into the model, thereby enabling the model to utilize the spatial correlation among the patches.

At the heart of the ViT architecture lies the transformer block, shown on the right side of Figure 1. It composed primarily of a Multi-Head Self-Attention (MHSA) layer and a Feed-Forward Neural Network (FFN) layer. The MHSA layer enables the model to discern the interaction and relationships between different patches, considering both their content and position. In its essence, MHSA is an extension of the self-attention mechanism, which calculates the dependencies between each element in the sequence and all the others. However, instead of performing this operation only once, multi-head attention does it multiple times, in parallel, for each ’head’. Each head learns and captures different types of relationships within the data, leading to better end classification accuracy.

For each attention head, the input patch embeddings are linearly projected into three distinct spaces, represented by Query (Q), Key (K), and Value (V) matrices. The purpose of these projections is to create different representational spaces that can learn to capture different aspects or features of the input data. At each head, we perform the following operation:

$$Attention(Q_i, K_i, V_i) = \text{softmax}\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right) V_i$$

Here, the softmax function is applied to ensure the weights are positive and sum to 1, d_k is the hidden dimension of the queries and keys, and the operation is done for the i-th attention head. The outputs of all the heads are then concatenated and linearly transformed to form the final output. This attention mechanism is followed by a layernorm, which we will delve in depth into in our experiments section. Lastly, the data is passed through an MLP with two fully-connected linear layers and an intervening ReLU activation. This last layer combines the "signal" from each of the heads into a final representation. The encoder block also includes two residual connections, shown by the side arrows on the right side of Figure 1, whose importance we will expound in our experiments section. Models typically have multiple Transformer Encoder block stacked on top of each other for deeper representations and performance.

Lastly, the classification head (MLP head on the top left side of Figure 1) is responsible for transforming the patch embeddings into final output predictions. Post-processing through numerous transformer blocks, the patch embeddings are fed into the classification head, a linear layer succeeded by a softmax activation function. Specifically, the class token, represented by the transformed embedding of the first patch, is processed through the linear layer. Subsequently, the softmax function generates class probabilities.

3.2. Model Alterations

We ran six experiments where we picked different aspects of the ViT architecture and altered them to values that we thought would make the ViT perform far worse than the baseline implementation. We did this because the ViT paper already achieves SoTA accuracy on various benchmarks, so instead of trying to beat its accuracy, we aimed to understand which parts of the ViT lead to its success and accuracy. By decreasing the quality of different aspects of the implementation in isolation, we were able to visualize and determine how each part of the vision transformer contributes to the saliency maps it learns. In addition to visualizations, we calculated various quantitative metrics that aided our understanding of the model's overall performance.

3.3. Visualization Approach

The goal of visualization is to see where on an image the ViT "concentrates" on when making predictions. This involves being able to see how attention flows from the start to the end of a Transformer. To quantify this, we used a technique called "Attention Rollout" in Quantifying Attention Flow in Transformers by Abnar and Zuidema [4]. Additionally, we took inspiration from the following blog post [7] and accompanying Github repo [6] to translate the method initially designed for NLP transformers for ViTs.

The method involves multiplying the attention by the gradient of the intended class output, then taking an average across all attention heads, disregarding the negative ones. This enables us to separate and maintain only the attentions that help delineate the target class (or classes). We obtain an attention Matrix A_{ij} at every Transformer block that demonstrates how much attention flows from token j in the preceding layer to token i in the succeeding one. We multiply these matrices between each pair of layers to understand the overall attention flow between them. However, we need to account for the residual connections - so we add the identity matrix I to the attention matrices at each layer: $A_{ij} + I$. Despite the Attention Rollout paper advocating for the mean of the heads, our empirical findings suggest that using the smallest value among attention heads yields more visually coherent and focused saliency maps. This intuitively aligns with the idea that multiple attention heads focus on various aspects, and using the minimum allows us to eliminate significant noise by pinpointing their shared focus. This provides us a method to iteratively calculate the Attention Rollout matrix at layer L:

$$AttentionRollout_L = (A_L + I) \cdot AttentionRollout_{L-1}$$

The last step in processing involves concentrating solely on the highest attentions, while discarding the others. This helps our saliency map to solely depict the Visual Transformer ViT focusing on particular sections of an image, which improves the way it singles out the object of interest.

3.4. SMM: A Novel Metric

Once we generated saliency maps for each experiment, we wanted a way of quantitatively analyzing them. We decided to construct a novel metric that we coined the "Saliency-map Mask Metric" (SMM). The calculation is intuitive: we compare the area of the image being attended to by the ViT to a mask of the object of importance. Thus, we are able to get a quantitative measure of how *accurate* the area is ViT attends to. This is done by using the Fast R-CNN [8] model to perform object instance segmentation on the input image, which outputs a mask over the image that contains the pixels of the object being classified. This mask is compared pixel-by-pixel to the visualization mask that is generated by our attention visualization from each image. Before this, a filter is applied to the visualization output that sets all pixel values to zero that are not above a pre-specified threshold (0.3). We then calculate the Jaccard similarity of the pixels that are in the segmentation with the pixels in the filtered attention mask.

The Jaccard similarity, often known as the Jaccard coefficient or the Intersection over Union (IoU), is a metric used to measure the similarity and diversity of sample sets. Mathematically, for two sets A and B, the Jaccard similarity J can be represented as:

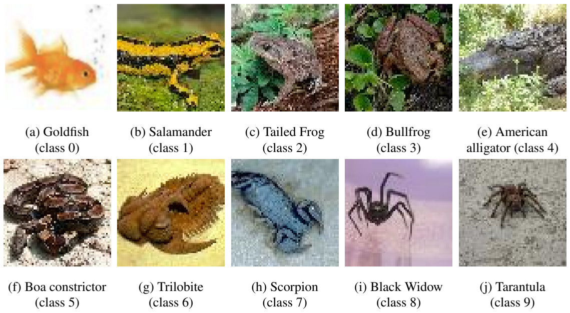
$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

The Jaccard similarity measures the overlap between the salient regions identified in the saliency map and the ground truth regions in the original image. A high Jaccard similarity implies that the saliency map effectively identifies critical regions in the image, while a low Jaccard similarity might indicate that the saliency map is highlighting irrelevant or less important parts of the image. This process is somewhat naive in that it only considers the pixels that are actually a part of the object being classified and not any pixels in its environment that may be useful in the classification that the model may be attending to. However, it still provides really interesting and useful information about how the ViT model is operating internally and how well it is attending to the important parts of the image.

4. Data

When deciding which dataset to choose, we preferred one with relatively few classes for ease of evaluating performance and producing visualizations. Following [12] which found ViTs typically need more data to reach the same accuracies of their CNN counterparts, we wanted our model to be able to adequately learn each class so that our saliency maps would be visually accurate as well as interesting. At the same time, we were wary to saturate the model with tens of thousands of images from each class as discussed above and needed to enforce a relatively low-data regime. Therefore, we decided to use the dataset Tiny Imagenet - introduced by Le et al in 2015 in ”Tiny Imagenet Visual Recognition Challenge.” [10] The dataset contains 100,000 colored images of 200 classes (500 for each class) downsized to 64x64 pixels. We chose 10 of the classes in the dataset and only use those 10 classes data to train and test the model (reasoning for limiting classes is aforementioned). Further, the 10 classes chosen–whose names and examples are visible in Figure 2–are particularly difficult to classify, as pairs of them are very similar looking. Notably, the goldfish shown will be the image that we run all our attention visualizations on to produce saliency maps. The remaining images highlight our chosen 10 classes. Of note is the extreme similarity in appearance between classes 2 and 3, classes 6 and 7 and classes 8 and 9. This will be useful to understand when we show the confusion matrices of our various models.

Each class has 500 training images, 50 validation images, and 50 test images. Each image was first converted into RGB format and then to a Pytorch tensor before being inputted into the model (and undergoing the patch embedding procedure as mentioned in the section above).



(a) Goldfish (class 0) (b) Salamander (class 1) (c) Tailed Frog (class 2) (d) Bullfrog (class 3) (e) American alligator (class 4)

(f) Boa constrictor (class 5) (g) Trilobite (class 6) (h) Scorpion (class 7) (i) Black Widow (class 8) (j) Tarantula (class 9)

Figure 2: Examples Each of the 10 Tiny Imagenet Classes Used

5. Experiments and Results

As mentioned before, all of our experiments were run on a ViT model as defined in the original paper [5] but implemented from scratch. Over the course of our experiments, we altered our positional encoding scheme, number of blocks in the encoder, number of attention heads, hidden dimension size, presence of residual connections, and presence of layernorm. For all other hyperparameters, we stuck with the values in the original ViT paper [5]: Adam with beta1 = 0.9, beta2 = 0.999, a high weight decay of 0.1, and a learning rate of 0.01 with a linear warm-up and decay. Unlike the paper, however, we used a batch size of 128 and patch size of 8 as both our dataset and image resolution were much smaller than the paper’s dataset (Imagenet). While an additional hyperparameter search fits other contexts, small improvements in performance would not affect our overall goal for this paper.

We trained each of the models for 50 epochs, and report quantitative and qualitative metrics including accuracy, recall, precision (shortened to ”prec” in the results tables), F1-score, ROC AUC, confusion matrix, and our saliency-map mask metric (SMM).

Overall, the greatest gains in performance above the baseline were using a learned positional encoding, fewer (2) attention heads and a higher hidden dimension (16). The two biggest detriments to performance were removing the residual connections and layernorms. Removing each of these in isolation caused accuracy to drop to *exactly* random performance (10%), meaning the model was incapable of learning anything without them. The following sections provide more detail on each experiment.

5.1. Baseline

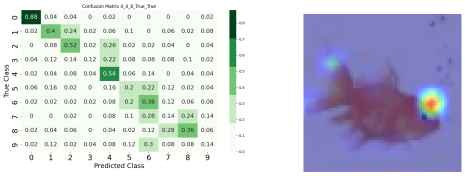
We define a baseline ViT model architecture in line with [5] that we then perform incremental alterations on. We use a 1D sine/cosine positional encoding, 4 blocks in the encoder, 4 attention heads, a hidden dimension of 8, residual connections, and layernorm. Table 1 summarizes these configurations. We see the results of the baseline experiment in Table 2, as well as the accompanying confusion matrix and saliency map of an image of a fish in Figure 3.

Positional Enc.	Encoder Blocks	Attention Heads	Hidden Dim	Residuals	Layernorm
Sin/Cos	4	4	8	Yes	Yes

Table 1: Baseline ViT Parameters

Train Acc	Test Acc	Recall	Prec	F1 Score	ROC AUC	SMM
0.476	0.328	0.314	0.367	0.312	0.766	0.025

Table 2: Baseline ViT Results on 10 Tiny Imagenet Classes



(a) Confusion Matrix

(b) Baseline

Figure 3: Confusion Matrix and Saliency Map for Baseline

Though the baseline accuracy is not above 50%, we contend this is still decent performance given only 500 training examples from each class. Further, when we consider the confusion matrix in Figure 3 and recognize the high similarity between adjacent classes shown in Figure 2, we see model's latent understanding of the classes is quite good: classes adjacent to the true class receive the bulk of incorrect labeling. We also notice the attention saliency map is focused on the fish's head near its eye.

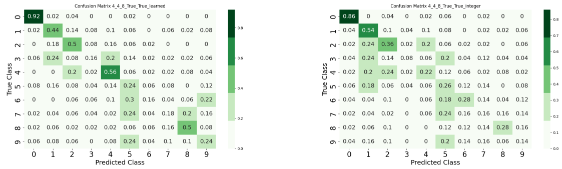
5.2. Positional Encoding

In our first experiment, we alter the default sine/cosine based positional encoding that are used in the classic Transformer architecture. In transformers, sine and cosine functions are used in the positional encoding give the model a sense of sequence order, something the self-attention mechanism doesn't inherently possess.

We first try a learnable positional encoding. Compared to the pre-defined, fixed sinusoidal approach the model has the freedom to learn the most suitable representation of position. ViTs break down images into non-overlapping patches, and the spatial relationship between these patches may be complex. By using a learnable positional encoding, the model can learn to adapt the encoding to the characteristics and spatial dependencies inherent in the image data, which could lead to better performance in image classification tasks. We also tried a simple integer positional encoding that simply assigned an incremented integer to each patch. We wanted to see how much it would negatively affect performance (which was our intuition). We see the results of this experiment in Table 3, as well as the accompanying confusion matrix in Figure 4 and saliency map of an image of a fish in Figure 5.

Pos Enc	Train Acc	Test Acc	Recall	Prec	F1 Score	ROC AUC	SMM
Base	0.476	0.328	0.317	0.367	0.312	0.766	0.025
Learn	**0.536**	**0.390**	**0.365**	**0.384**	**0.376**	**0.802**	**0.065**
Integer	0.476	0.318	0.302	0.325	0.304	0.769	0.026

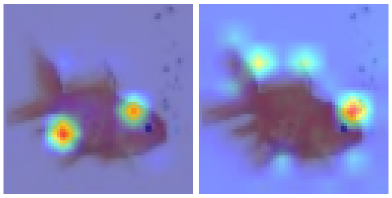
Table 3: ViT with Different Positional Encoding Results on 10 Tiny Imagenet Classes



(a) Learned Pos Embedding

(b) Integer Pos Embedding

Figure 4: Confusion Matrices for Positional Encoding Change



(a) Learned Pos Embedding

(b) Integer Pos Embedding

Figure 5: Saliency Maps for Positional Encoding Change

From the Table 3, learned positional encoding is the clear winner, achieving 39% test accuracy versus the 32.8% of the baseline and higher scores for all other methods. Further, the integer strategy achieved worse accuracy with 31.8% than the baseline, but was much closer than the difference between the baseline and the learned strategy.

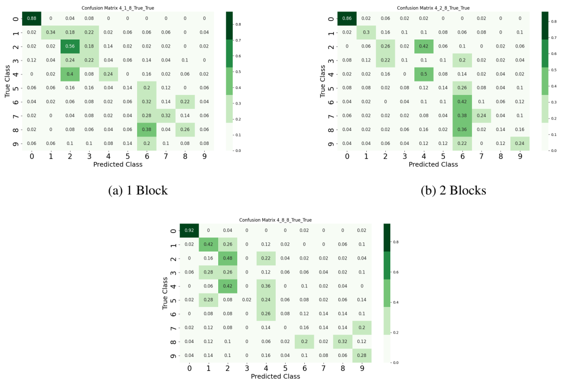
The generated saliency maps provide an interesting qualitative interpretation: the inferior integer strategy focuses on seven distinct positions around the fish. On the other hand, the dominant learned strategy focuses on just two spots on the fish. One interpretation of this is the better model is more focused on two important points than being scattered on many points.

5.3. Encoder Blocks

In our second experiment, we altered the number of transformer encoder blocks in the ViT encoder. As touched on earlier and shown in Figure 1, the ViT encoder is composed of several identical encoder blocks, each containing the multi-headed self-attention mechanism described in Section 3. Our intuition was that adding / removing blocks would boost / decrease the model's expressibility and thus influence performance. The baseline number of blocks was 4. We see the results of the experiment in Table 4, as well as the accompanying confusion matrix in Figure 6 and saliency map of an image of a fish in Figure 7.

Blocks	Train Acc	Test Acc	Recall	Prec	F1 Score	ROC AUC	SMM
1	0.429	**0.336**	**0.332**	0.348	**0.325**	0.747	0.009
2	0.475	0.320	0.309	0.323	0.306	0.751	**0.039**
Base	**0.476**	0.317	0.328	**0.367**	0.312	**0.766**	0.025
8	0.374	0.310	0.309	0.279	0.280	0.739	0.017

Table 4: ViT with Different Number of Blocks Results on 10 Tiny Imagenet Classes

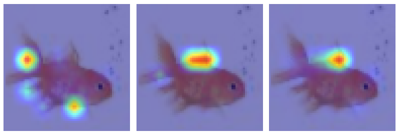


(a) 1 Block

(b) 2 Blocks

(c) 8 Block

Figure 6: Confusion Matrix for Number of Blocks in Encoder



(a) 1 Block (b) 2 Blocks (c) 8 Blocks

Figure 7: Saliency Maps for Number of Blocks in Encoder

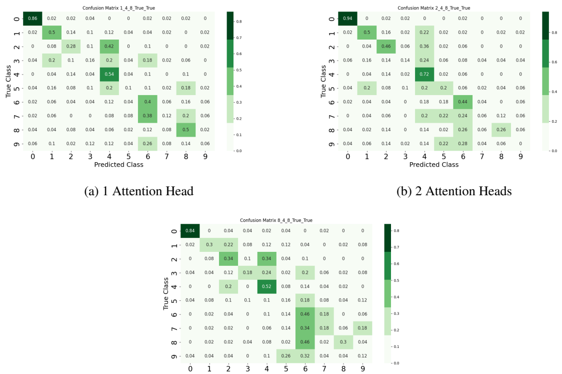
Unlike our intuition, the results show that the number of encoding did *not* change performance meaningfully. In fact, having only 1 encoding block achieved slightly better test accuracy (33.6%) than the baseline of 4 block (32.8%). The 1 block strategy also achieved better recall and F1 score than the baseline, that by small margins. Having 8 blocks, also degraded performance from baseline slightly. In summary, we believe the number of encoding blocks had minimal influence on performance because our training dataset was so small and did not allow for deeper representations to be learned. This hypothesis could be tested by training the same models on 10-100x more data.

5.4. Attention Heads

In our third experiment, we altered the number of attention heads in the multi-head attention module - whose purpose we thoroughly explained in our Methods section. We see the results of the experiment in Table 5, as well as the accompanying confusion matrix in Figure 8 and saliency map of an image of a fish in Figure 9.

Heads	Train Acc	Test Acc	Recall	Prec	F1 Score	ROC AUC	SMM
1	0.475	0.352	0.341	0.337	0.324	0.771	**0.041**
2	0.460	**0.372**	**0.370**	0.348	0.335	0.771	0.001
Base	0.476	0.328	0.317	0.367	0.312	0.766	0.025
8	**0.523**	0.340	0.321	**0.373**	**0.337**	**0.777**	0.029

Table 5: ViT with Different Number of Attention Heads Results on 10 Tiny Imagenet Classes

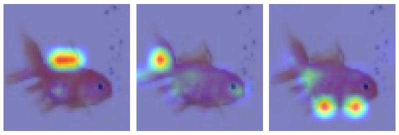


(a) 1 Attention Head

(b) 2 Attention Heads

(c) 8 Attention Heads

Figure 8: Confusion Matrix for Number of Attention Heads



(a) 1 Attention Head (b) 2 Attention Heads (c) 8 Attention Heads

Figure 9: Saliency Maps for Number of Attention Heads

The results of this experiment show a meaningful increase in test accuracy by *decreasing* the number of heads from 4 (Base) to 2 followed by a second best overall test accuracy of 1 head. We hypothesize that, similar to the preceding section, the training set may be too sparse to learn rich representations across many heads. This theory must be tested further.

The precision, recall, F1 and ROC AUC scores are marginally different between models. Notably, our novel SMM score is highest on the model with 1 head because, as Figure 9 shows, it is the model with the most attention overlaid on the body of the fish.

5.5. Hidden Dimension

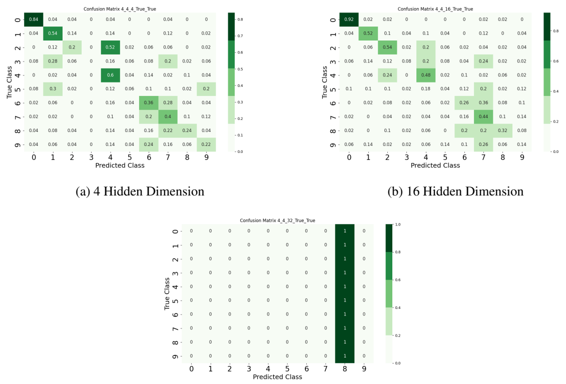
In our fourth experiment, we altered the hidden dimension size. The hidden dimension in Vision Transformers (ViTs) refers to the dimensionality of the internal representations that the model learns. After the input image is divided into patches and each patch is flattened, these flattened patches are linearly transformed into embeddings of this hidden dimension. Throughout the transformer's layers, all the operations (self-attention, feed-forward neural networks) operate on these embeddings and maintain this dimension. The choice of hidden dimension is a trade-off between computational resources and model capacity. A larger hidden dimension size can increase the model's capacity, allowing it to learn more complex representations and potentially improving performance. However, this comes at the cost of increased computational requirements, both in terms of memory and processing power. Conversely, a smaller hidden dimension size reduces computational demands but may limit the complexity of the representations the model can learn, potentially resulting in poorer performance.

We see the results of the experiment in Table 6, as well as the accompanying confusion matrix in Figure 10 and saliency map of an image of a fish in Figure 11.

Hid Dim	Train Acc	Test Acc	Recall	Prec	F1 Score	ROC AUC	SMM
4	0.388	0.346	0.342	0.311	0.310	0.735	0.021
Base	0.476	0.328	0.317	**0.367**	0.312	0.766	0.025
16	**0.510**	**0.374**	**0.369**	0.356	**0.348**	**0.789**	**0.026**
32	0.100	0.100	0.100	0.010	0.018	0.502	0.017

Table 6: ViT with Different Hidden Dimension Size Results on 10 Tiny Imagenet Classes

The results of this experiment are surprising. As we expected, a higher hidden dimension (16) performs better than the base model (dim 8). This was not, however, the case for a hidden dimension of 32, which performed worse of all and simply guessed class 8 for all images. We suspect this is because the model overfit in this situation, due to there being too many learned parameters and not enough training images (only 500 per class with some classes having similar images).

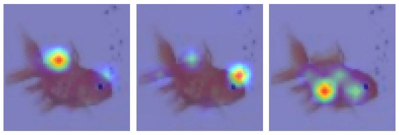


(a) 4 Hidden Dimension

(b) 16 Hidden Dimension

(c) 32 Hidden Dimension

Figure 10: Confusion Matrix for Hidden Dimension



(a) Size-4 Hidden Dimension

(b) Size-16 Hidden Dimension

(c) Size-32 Hidden Dimension

Figure 11: Saliency Maps for Hidden Dimension

5.6. Residual Connections

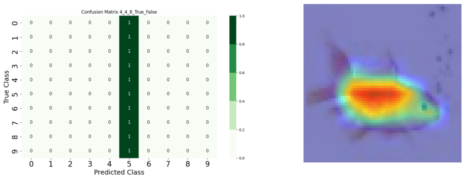
In our fifth experiment, we tried removing all residual connections from the encoder in the ViT. In the baseline implementation, residual connections are used in order to increase model performance by preventing overfitting as well as reducing the effect of the vanishing gradient problem. They enable the direct flow of information by adding the original input to the output of a layer, creating a shortcut connection. We see the results of the experiment in Table 7, as well as the accompanying confusion matrix and saliency map of an image of a fish in Figure 12.

Res-Conns	Train Acc	Test Acc	Recall	Prec	F1 Score	ROC AUC	SMM
Base	**0.476**	**0.328**	**0.317**	**0.367**	**0.312**	**0.766**	0.025
No	0.100	0.100	0.100	0.010	0.018	0.500	**0.297**

Table 7: ViT with and without Residual Connections Results on 10 Tiny Imagenet Classes

As expected, the model performs no better than random with 10% test accuracy. Interestingly, it guesses class 5 for all the classes, which is likely because removing the residual causes the model to struggle with propagating the gradient back through the many layers during training, often referred to as the vanishing gradient problem. In such a scenario, earlier layers of the model learn very slowly or stop learning altogether, leading to sub-optimal performance.

Further, the uniform prediction for class 5 could be a



(a) Confusion Matrix

(b) Without Residual Connections

Figure 12: Confusion Matrix and Saliency Map for No Residual Connections

symptom of the model failing to differentiate the input features across different classes. Without residual connections, the model might be falling into a simplistic strategy of predicting the most common or easiest-to-distinguish class (boa constrictor in this case) instead of learning meaningful patterns from the data. This implies that the complexity and depth offered by the model architecture are being completely lost without the residual connections.

Though the saliency map looks very good at first blush, we suspect this was randomly fortunate because the fish happens to be in the center of the image, which overlaps with the randomly initialized attentions. Further, since these random attentions covers a wide area of each image, this likely boosted the SMM score above the baseline. This highlights why the SMM score must be used with other metrics to gain a complete picture of model performance.

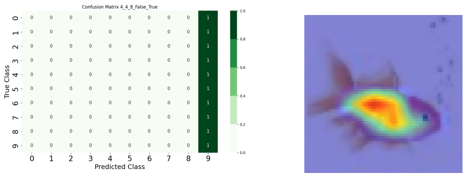
5.7. Layernorm

In our final experiment, we removed the layernorm layers from each encoder block in the ViT. Layer normalization is a crucial component in ViTs that helps stabilize the learning process and improve model performance. By normalizing the inputs across the hidden dimension, layer normalization reduces the impact of varying scales and offsets in the data, making the model more robust and less sensitive to parameter initialization. We see the results of the experiment in Table 8, as well as the accompanying confusion matrix in Figure and saliency map of an image of a fish in Figure 13.

Layer-norm	Train Acc	Test Acc	Recall	Prec	F1 Score	ROC AUC	SMM
Base	**0.476**	**0.328**	**0.317**	**0.367**	**0.312**	**0.766**	0.025
No	0.100	0.100	0.100	0.010	0.018	0.500	**0.482**

Table 8: ViT with and without Layernorm Results on 10 Tiny Imagenet Classes

The results are very similar to the preceding section where removing the layernorm strips the model of all of its expressibility and ability to learn complex relationships between images. In this case, the model outputs class 9 for every input. We think the removal of layer normalization led to unstable gradients during training, causing the model to become stuck and not learn effectively. Without layer normalization, the scales and distributions of activations throughout the model can vary wildly during training, leading to gradient explosions or vanishing gradients. Furthermore, the removal of layer normalization may increase the model's sensitivity to the initial parameter values. This could lead to the model converging to sub-optimal solutions that are heavily influenced by its initial state. This experiment shows that layer normalization is essential for maintaining stability during training and allowing the model to learn complex patterns within the data.



(a) Confusion Matrix

(b) Without Layernorm

Figure 13: Confusion Matrix and Saliency Map for No Layernorm

Like the preceding section, the saliency map is random and not informative to the function of the model.

6. Conclusion/Future Work

In conclusion, our methods for visualizing and analyzing the performance of vision transformers, our careful selection of experiments to modify our custom ViT implementation and our deliberate choice to limit the amount of data each model trained on allowed us to understand and present both what attention is ”looking at” and how important each component of a ViT is.

Of all architectural changes explored–positional encoding, number of encoder blocks, number of attention heads, size of the hidden dimension and the presence of residuals and layernorms–on our ViT model, a learned positional encoding and a hidden size of 16 boosted performance the most. In this data constrained environment, with only 500 32 x 32 images from 10 classes of the Tiny Imagenet dataset [10], we showed that modulating the number of encoding blocks and attention heads had marginal affect on performance. Lastly, we demonstrated the absolute necessity of layernorm and residual connections in ViT models.

Besides accuracy and the standard quantitative metrics, we judged our model using two novel ideas: attention visualizations and the Saliency-Map Mask Metric (SMM). The attention saliency maps gives a qualitative interpretation of performance. We then introduced the SMM metric, which compares the overlap between the attention and the ground truth object segmentation, as a quantitative measure. We demonstrate that both metrics cannot be used in isolation and must be used together for best interpretability.

In the future, we would continue to run ablation on different parts of the ViT. Once we better understood the best architecture for this low-data environment, we would collect 10 to 100 times more training images from each class and re-run all experiments in this non-constrained environment. We would be interested if and how these optimal model configurations changed in this new data regime.

7. Contributions / Acknowledgements

Note: For most of the project, we worked as a group coding together, so there is a lot of overlap for what we all did.

Aayush: worked on image visualization code and experiment design and execution, as well as modifying the Vision Transformer for experiments, spearheaded the paper

Elliot: worked on model training and instance segmentation / calculation of the SMM metric, worked on visualization code, wrote the paper intro and related works

Finn: implemented the Vision Transformer from scratch and modified it for the experiments, worked on image visualization code, worked on the paper's methods section

For the instance segmentation to compute our SMM metric, we modified code from this GitHub repo: `https://github.com/spmallick/learnopencv/tree/master/PyTorch-Mask-RCNN`.

For the attention map visualization, we modified code from this GitHub repo: `https://github.com/jacobgil/vit-explain`.

Additionally, the following is a link to our code: `https://github.com/finndayton/CS231N-Final-Project`.

References

[1] pytorch Python library.

[2] scikit-learn Python library.

[3] matplotlib Python library.

[4] S. Abnar and W. H. Zuidema. Quantifying attention flow in transformers. *CoRR*, abs/2005.00928, 2020.

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[6] J. Gildenblat. Awesome machine learning. GitHub Repository, 2021. Retrieved from `https://github.com/jacobgil/vit-explain/tree/15a81d355a5aa6128ea4e71bbd56c28888d0f33b`.

[7] J. Gildenblat. Exploring explainability for vision transformers. Blog post, 2021.

[8] R. Girshick. Fast r-cnn, 2015.

[9] T. Klug and R. Heckel. Scaling laws for deep learning based image reconstruction. In *The Eleventh International Conference on Learning Representations*, 2023.

[10] Y. Le and X. Yang. Tiny imagenet visual recognition challenge. *Stanford University, CS231N*, 2015.

[11] M. Raghu, T. Unterthiner, S. Kornblith, C. Zhang, and A. Dosovitskiy. Do vision transformers see like convolutional neural networks?, 2022.

[12] H. Wu, B. Xiao, N. Codella, M. Liu, X. Dai, L. Yuan, and L. Zhang. Cvt: Introducing convolutions to vision transformers, 2021.