Comparing Vision Transformers (ViTs) and CNNs on FashionMNIST and CIFAR100

Alex Wa, Derek Gao, Jack Yin, Andy Xu

May 3, 2025

1 Introduction

Convolutional Neural Networks (CNNs) have been the dominant architecture in computer vision for decades, using convolutional layers to extract spatial features. Vision transformers (ViTs) were recently introduced (Dosovitskiy et al., 2020), relying on self-attention mechanisms instead of convolutions.

CNNs are designed with strong spatial inductive biases (e.g. translational invariance and hierarchical feature learning), often generalizing well even on limited data. In contrast, ViTs rely more on data size and augmentations to learn useful representations, which may raise concerns about applicability to scenarios where labeled data is scarce. Hybrid models like Swin Transformer (Liu et al., 2021) and CvT (Wu et al., 2021) bridge the gaps by combining feature extraction with long-range dependencies.

Through a series of experiments, we seek to understand how ViT performance compares to that of CNNs when trained from scratch on smaller-sized datasets. As a corollary, we may also wish to evaluate the hybrid architectures. Finally, we want to understand how the two architectures differ based on computational efficiency.

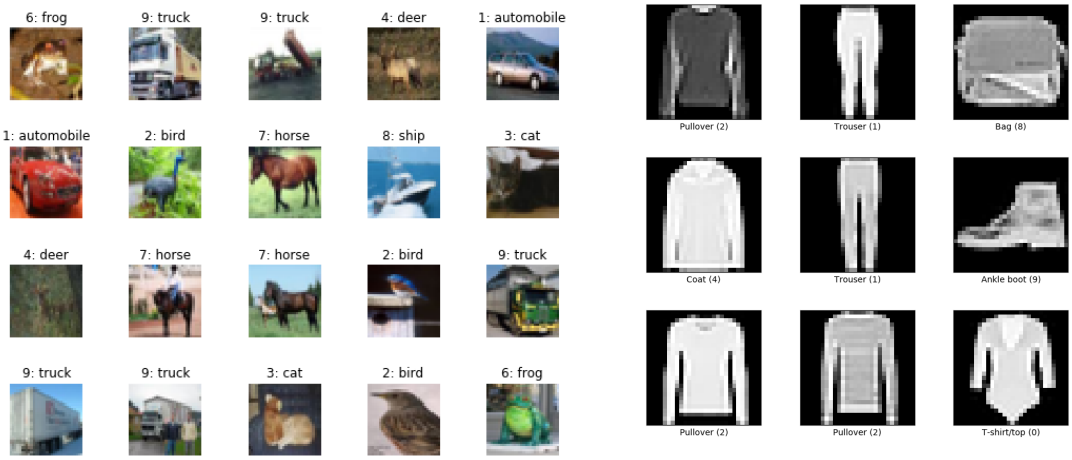
2 Data

We trained and tested our models using two well-known datasets: CIFAR-100 and FashionMNIST. They are two of the most widely used small-scale benchmarks in computer vision, each providing a method to evaluate model capability and robustness.

CIFAR-100 is composed of 60,000 color images of size 32×32 pixels. These images are divided into 100 fine-grained classes and grouped into 20 super-classes, with 50,000 training and 10,000 testing images. CIFAR-100's low spatial resolution forces models to distinguish between subtle inter-class distinctions, such as different animal species of types of vehicles. Its images are small, but with a large number of classes, CIFAR-100 presents itself as a medium-sized dataset ideal for stresstesting for architectures that must balance training accuracy with overfitting risk.

By contrast, FashionMNIST contains 70,000 grayscale images of wardrobe items, each 28x28 pixels, across 10 broad categories such as "T-shirt/top" or "Bag". FashionMNIST is split into 60,000 training and 10,000 testing images and contains both variation within classes and ambiguity across classes. This forces models to capture both local details and larger global patterns. FashionMNIST has become a standard sanity check dataset for new architectures: an accuracy score below 85 or 90% is often an indication that a model may need corrections before tackling more complex tasks. A sample from each dataset is shown in Figure 1.

CNNs look at small patches of an image at a time before combining those features into higher-level patterns, thus making them a natural fit (fast and reliable) for both datasets. ViTs take a very different and novel approach, as proposed by (Dosovitskiy et al., 2020). They break each image into a sequence of patches (for example 4x4 pixels), usually ordered from top-down and left-to-right. The architecture then turn those patches into tokens, and then use self-attention to learn how every part relates to others. Without the built-in locality of CNNs, however, ViTs can struggle on smaller datasets and often require more regularization. Given the transformer architecture, ViTs are also more costly to compute. For



CIFAR-100 FashionMNIST

Figure 1: Sample CIFAR-100 and FashionMNIST Data

example, the models presented in (Dosovitskiy et al., 2020) have 86M, 307M, and 632M parameters for the base, large, and huge models, respectively. Their huge model used 2.5k TPUv3-core-days to train on their in-house JFT-300M dataset and 230 TPUv3-core-days to train on ImageNet.

Together, FashionMNIST and CIFAR-100 allow for testing models in stages: the former is simple enough for quick debugging and hyperparameter tuning, while the latter offers a more rigorous test that pushes a model's ability to learn fine-grained, low-resolution images without overfitting. By training our implementations on both datasets, we were able to gain essential insights into the performance of ViTs and whether they can surpass CNNs in practical, small-scale vision tasks.

3 Methodology

We conducted a comparative analysis between CNNs and ViTs, training each architecture from scratch on the two datasets FashionMNIST and CIFAR-100. We then assessed model generalization under constrained data regimes and evaluated computational tradeoffs.

For both datasets, we augmented the training set with random horizontal flips and color jittering. However, the pre-processing pipline diverges from there. Post-augmentation in FashionMNIST, we applied grayscale normalization. For CIFAR-100, we normalized all images using CIFAR-100's specific mean and standard deviation values. We then loaded data using PyTorch's DataLoader using batch sizes typically set to 64 or 128.

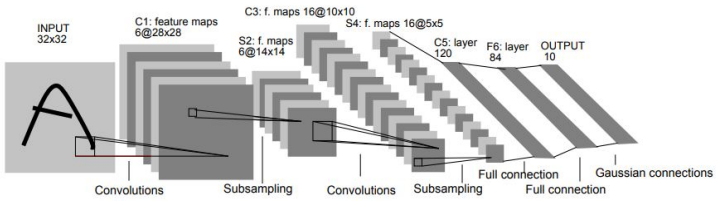


Figure 2: LeNet architecture for 32×32 images

For our experimentation in regards to the CNNs, we tried classic CNN models such as LeNet (Figure 2) and ResNet variants (Figure 3), particularly ResNet-18. For LeNet, we tried various activation and pooling combinations, trying out ReLU, TanH, and GELU in combination with Max and Average

respectively. For ResNet-18, we also trained for deeper feature hierarchies, especially on CIFAR-100.

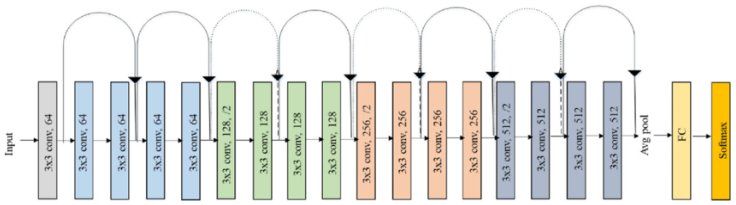


Figure 3: Resnet-18 architecture

We trained each model with cross-entropy loss and the Adam optimizer, tracking performance using training and validation accuracy and loss over 25 epochs (in some places, we had modified it to 30 epochs). We also tracked additional metrics including F1-score, precision, and recall to evaluate our results, though we were more interested in the aforementioned accuracy and losses.

For our experimentation in regards to Vision Transformers, we followed the structure proposed in Dosovitskiy's article (Figure 4). We did this by patch embedding the images into 1D token sequences, using a transformer encoder stack with multi-head self-attention and layer normalization, and using a MLP head for final classification.

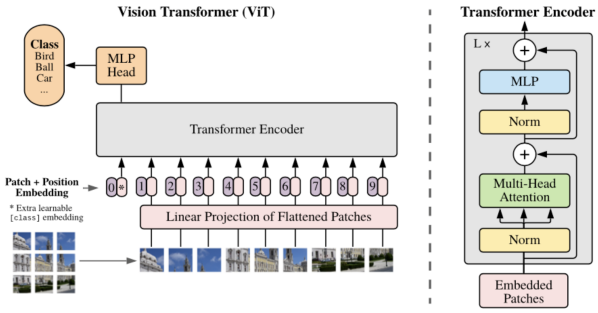


Figure 4: ViT architecture

We also implemented GELU activations, multi head attention, and Transformer encoder blocks, and then trained the models using similar loss functions and optimizers as for the CNNs. Where feasible, we also used comparable hyperparameters for comparison. We tracked metrics such as attention maps, accuracy, and loss throughout the training to analyze convergence and performance stability results.

As a final note, we used PyTorch, torchvision, sklearn, and wandb frameworks while training all models with CUDA acceleration when available, and the runtime per training run ranged from 1 to 10 minutes depending on model size, dataset, and training method.

4 Implementation Details

We balanced three main considerations when choosing our hyperparameters for our models: model capacity, training stability, and overfitting risks.

We used a batch size of 128 throughout because it was large enough to give stable gradient estimates but small enough to fit comfortably on most GPUs. For CNNs, we used the standard stochastic gradient descent plus momentum 0.9 and learning rate 0.1, a combination of parameters known to work well on small image datasets by allowing smooth updates and accelerating convergence. A weight-decay of $5 \cdot 10^{-4}$ regularized the hundreds of thousands of CNN parameters to prevent overfitting, especially because FashionMNIST included 60,000 training samples. On the other hand, ViTs are more sensitive to step-size and regularization, so we switched to AdamW and started with a smaller learning rate ($1 \cdot 10^{-3}$ or $5 \cdot 10^{-4}$) to handle high-dimensional parameter spaces, like self-attention layers, and prevent the model from memorizing patterns in the training data. We also apply a cosine-annealing schedule over 25 epochs

to gradually lower the learning rate, allowing the model to refine representations.

Because FashionMNIST is relatively simple, we found that hidden sizes (the width of each token vector) of 32 and 48 in our ViTs were big enough to capture the shape and texture cues that distinguished between similar clothing items, like a sneaker and an ankle-boot. Four layers were enough to let the model build progressively from local patch features to global shape cues, while four heads, each with an 8-12-dimensional subspace, balanced diverse attention patterns and manageable parameter counts. Finally, training our model over 40 epochs was sufficient to allow the data to fully converge.

On the other hand, CIFAR-100 contains 100 fine-grained classes and color images, so we needed more capacity and depth. Thus we chose wider hidden sizes: 128 and 256, so that each token vector could encode additional, richer information. We chose 128 to keep training time reasonable, and 256 when we wanted to push the accuracy higher and potentially accept overfitting risks. We then enhanced our model to 12 layers and 8 heads, giving each head a 16-32-dimensional attention subspace. This way, each head was able to capture the subtle relationships between patches. Our learning rates and dropout were constant across both datasets, but we extended training to 150 epochs so that this deeper model could fully learn the 50,000 training images without memorizing them.

5 Results

5.1 FashionMNIST

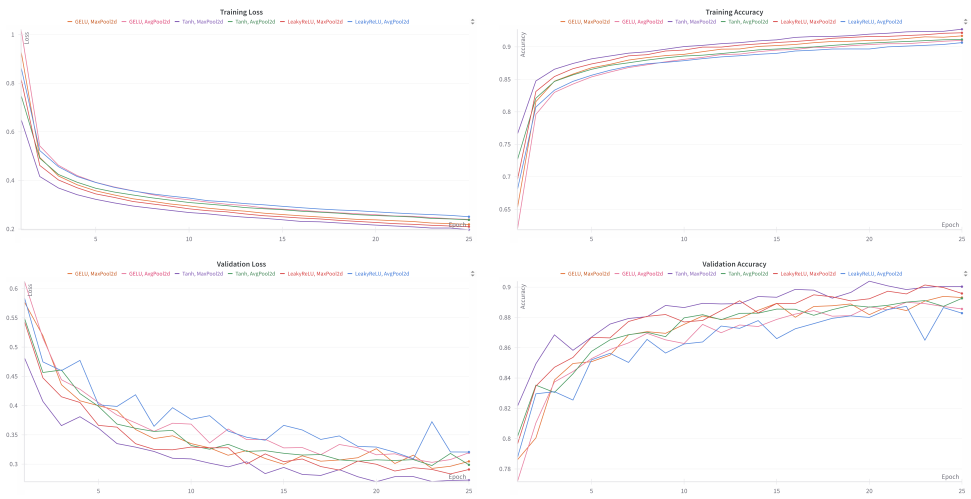


Figure 5: Training/Validation Performance Across Activation and Pooling Configurations for FashionMNIST using LeNet architecture

Method	Validation Loss	Validation Accuracy	Training Loss	Training Accuracy
GELU, Max	0.305	89.31%	0.218	94.83%
GELU, Avg	0.320	88.56%	0.239	91.99%
LeakyReLU, Max	0.291	89.58%	0.209	94.03%
LeakyReLU, Avg	0.321	88.28%	0.251	92.07%
Tanh, Max	0.273	90.03%	0.197	95.15%
Tanh, Avg	0.299	89.26%	0.238	92.83%

In Figure 5, we empirically observe the benefit of using MaxPool2d over AvgPool2d, and Tanh over other activation functions like LeakyReLU and GeLU. This makes sense because the background of FashionMNIST is black, and MaxPool2d selects the brighter, more relenvant pixels. The end results are summarized in the table above, where the LeNet architecture using Tanh and MaxPool2d beats models with other hyperparameters across all four metrics after training for 25 epochs.

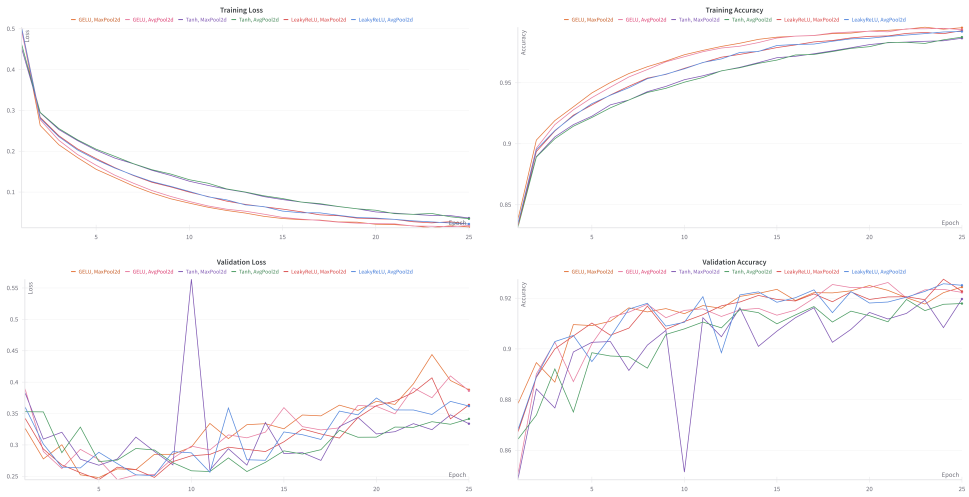


Figure 6: Training/Validation Performance Across Activation and Pooling Configurations for FashionMNIST using ResNet architecture

Method	Validation Loss	Validation Accuracy	Training Loss	Training Accuracy
GELU, Max	0.388	92.44%	0.015	99.52%
GELU, Avg	0.386	92.23%	0.018	99.36%
LeakyReLU, Max	0.364	92.27%	0.021	99.27%
LeakyReLU, Avg	0.362	92.51%	0.022	99.21%
Tanh, Max	0.334	91.97%	0.036	98.65%
Tanh, Avg	0.342	91.79%	0.035	98.74%

In Figure 6, the benefit of a specific activation function or a pooling method is much less clear here when using the ResNet-18 architecture , as all methods besides the two models using Tanh. However, we do observe the classic bias-variance tradeoff phenomenon when looking at the validation loss, which decreases from epochs 0 to 8, then increases gradually from epochs 9 to 25. Still, the validation accuracy increases by roughly .50% between epoch 9 and 25 for all six models. This reinforces the idea that in ResNet-18, architectural priors like skip connections dominate, and the network's depth and design may compensate for suboptimal activation-pooling pairings.

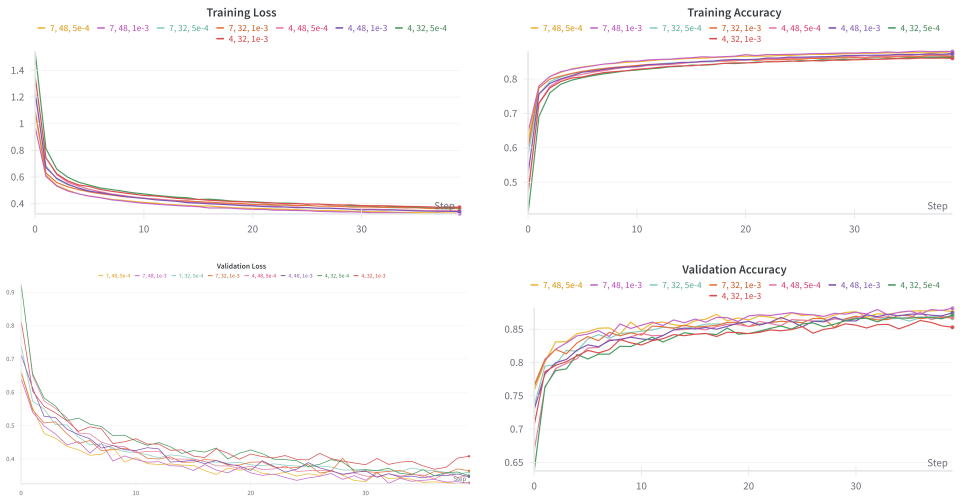


Figure 7: Training/Validation Performance Across Patch Size, Hidden size, and Learning Rate for FashionMNIST using ViT architecture

From Figure 7 and the table, we find that on average, using the larger patch size of 7 (1/4th of the image size) leads to higher validation accuracies. This might be because of the spatial granularity that helps the ViT extract more semantically meaningful features early on. Another reason could be the sequence length and more stable self-attention patterns. Moreover, the larger hidden size of 48, as compared to the hidden size of 32, also increased. As a result of the larger hidden size, the learning rate does not contribute much to the difference.

Method (ps, hs, lr)	Validation Loss	Validation Accuracy	Training Loss	Training Accuracy
7, 48, 5e-4	0.329	87.62%	0.327	87.84%
7, 48, 1e-3	0.347	88.16%	0.324	88.01%
7, 32, 5e-4	0.361	86.64%	0.364	86.62%
7, 32, 1e-3	0.365	86.76%	0.366	86.50%
4, 48, 5e-4	0.408	86.88%	0.348	87.00%
4, 48, 1e-3	0.347	87.46%	0.340	87.39%
4, 32, 5e-4	0.350	87.14%	0.370	86.28%
4, 32, 1e-3	0.365	85.29%	0.374	86.00%

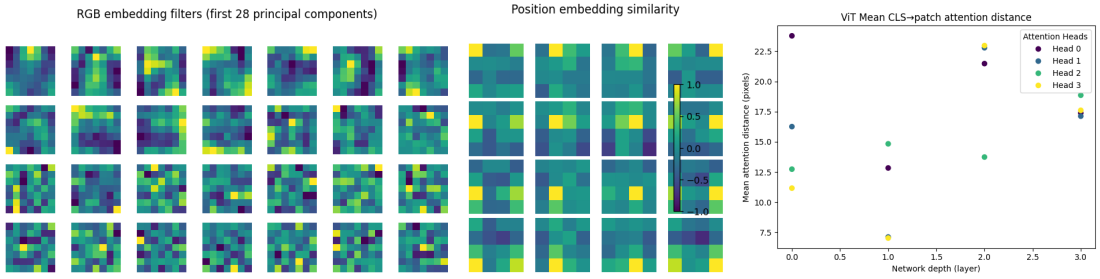


Figure 8: **Left:** filters of the initial linear embedding of values. **Center:** Similarity of position embeddings. **Right:** Size of attended area by head and network depth.

In Figure 8, we visualize the attention mechanisms of the ViT, specifically how the pixel patches are projected onto the input space, how the model distinguishes different image regions, and how attention shifts between the layers. In the middle, we confirm that the model is correctly learning the positional encoding relative to any another. Lastly, we see indeed, the ViT is referencing pixels, on average, 15 pixels away, whereas the classic CNN would only reference around the convolutional window.

5.2 CIFAR-100

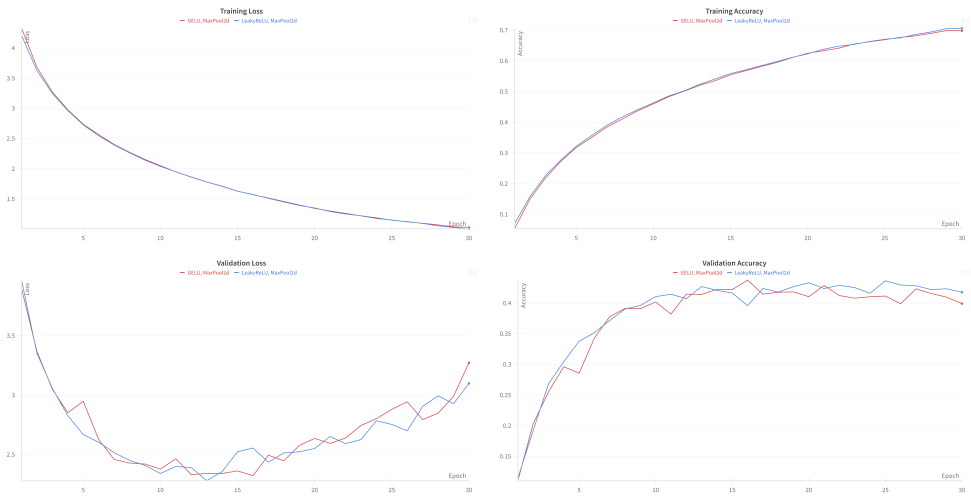


Figure 9: Training/Validation Performances for CIFAR100 using LeNet architecture

Unfortunately, CIFAR-100 gave us a lot of trouble when training, as observed between the discrepancy between the validation and training accuracy. In Figure 9, we again observe the bias-variance tradeoff in the validation loss, which turns around epoch 12. The validation accuracy remained relatively constant between epoch 12 and 30.

Method	Validation Loss	Validation Accuracy	Training Loss	Training Accuracy
GELU, Max	3.271	39.91%	1.030	69.81%
LeakyReLU, Max	3.098	41.76%	1.008	70.64%

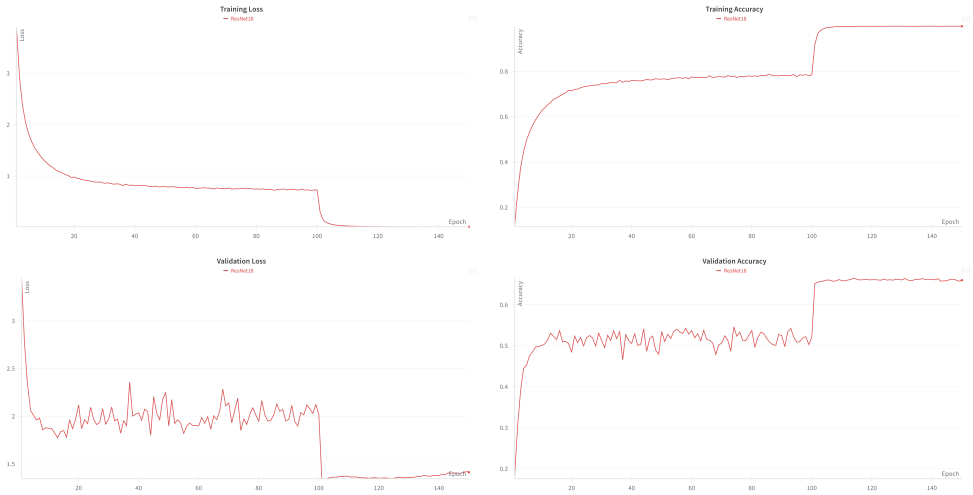


Figure 10: Training/Validation Performance for CIFAR100 using ResNet architecture

Fascinatingly, we observe grokking in Figure 10: after many epochs of slow progress, the model suddenly improves in both training and validation accuracy around epoch 105. This sharp drop in loss and jump in accuracy suggests a shift from memorization to generalization. The delayed generalization may stem from the model first learning superficial patterns before discovering the deeper structure, likely caused by ResNet's architecture and inductive biases common to many CNNs.

Method	Validation Loss	Validation Accuracy	Training Loss	Training Accuracy
ResNet18	1.414	66.01%	0.011	99.97%

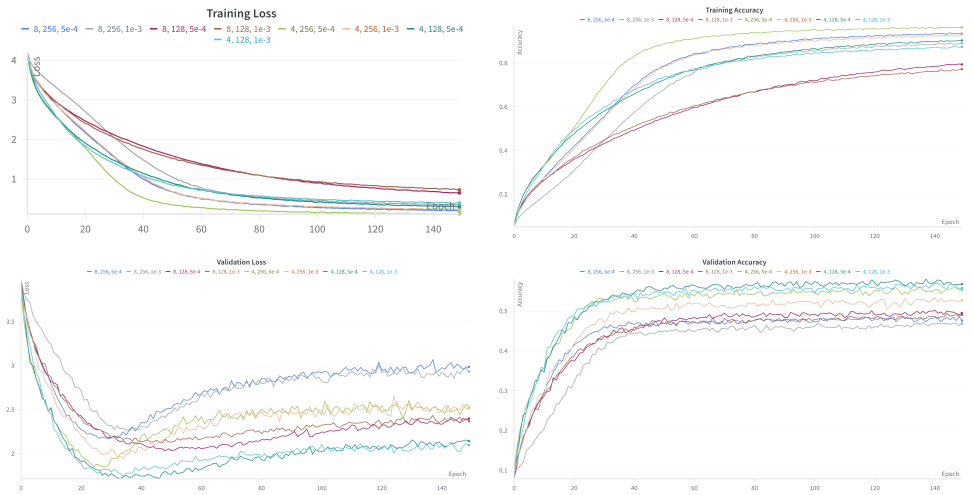


Figure 11: Training/Validation Performance for CIFAR100 using ViT architecture

Figure 11 shows that while all ViT models converge in training, only certain configurations generalize well to CIFAR100, highlighting the sensitivity of transformer models to hyperparameter choices. The

patch size of 4 consistently yield better validation performance, which differs from the larger patch size of 7 for FashionMNIST. This is likely due to noisier gradient estimates acting as a form of implicit regularization that helps prevent overfitting. Models with a hidden size of 128 outperform those with 256, which may be because excessive capacity may lead to memorization rather than generalization on a dataset like CIFAR100. Learning rates around 5e-4 to 1e-3 appear optimal, as lower rates slow convergence and higher rates can destabilize training, especially when paired with large hidden sizes.

Method (ps, hs, lr)	Validation Loss	Validation Accuracy	Training Loss	Training Accuracy
4, 128, 5e-4	2.142	56.81%	0.299	90.32%
4, 128, 1e-3	2.096	55.77%	0.393	87.35%
4, 256, 5e-4	2.525	55.26%	0.207	96.29%
4, 256, 1e-3	2.516	52.68%	0.223	93.02%
8, 128, 5e-4	2.399	49.52%	0.652	79.36%
8, 128, 1e-3	2.370	49.06%	0.727	77.19%
8, 256, 5e-4	2.988	47.63%	0.207	93.41%
8, 256, 1e-3	2.932	46.86%	0.345	89.02%

6 Acknowledgments

We would like to thank Professor Alex Wong for allowing us to use his compute and for his thoughtfulness in teaching CPSC381 this semester. Every day is a beautiful day to learn ML!

References

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, *abs/2010.11929*. Retrieved from https://arxiv.org/abs/2010.11929

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., ... Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. *CoRR*, *abs/2103.14030*. Retrieved from https://arxiv.org/abs/2103.14030

Wu, H., Xiao, B., Codella, N., Liu, M., Dai, X., Yuan, L., & Zhang, L. (2021). Cvt: Introducing convolutions to vision transformers. *CoRR*, *abs/2103.15808*. Retrieved from https://arxiv.org/abs/2103.15808

7 Supplementary Materials

The Github repository containing code and any necessary instructions and reproducibility assets may be found here.