Evaluating the Evolution of Deep Learning Techniques for Image Classification by Team DALISA

David Exiga
edavid9@gatech.edu

Ali Jalilian
jalilian@gatech.edu

Sara Bahri
sbahri7@gatech.edu

Abstract

This paper presents a systematic study comparing convolutional neural networks, vision transformers, and diffusion based data augmentation for image classification. As part of our investigation, we conduct a detailed hyperparameter optimization study for training ResNet-32 on the CIFAR-10 dataset. We explore the effects of different optimizers (SGD with momentum vs. Adam), learning rates, weight decay values, and learning rate schedulers on model performance. Our experiments demonstrate that SGD with a learning rate of 0.1, weight decay of 1×10^{-4}, and a multi-step learning rate schedule achieves the best validation accuracy of 87.50% after 20 epochs of training. We provide detailed analysis of training dynamics and discuss the trade-offs between different hyperparameter configurations. In addition to the CNN baseline, we implemented and evaluated several Vision Transformer (ViT) models of varying depth and parameter count on CIFAR-10 to better understand how attention based architectures behave under limited data and compute. Our experiments show that while ViTs can achieve competitive accuracy with properly chosen patch sizes and moderate model depth, deeper models tend to overfit severely and underperform unless trained with significantly more data and computational resources. Finally, using DiffuseMix to augment the data proved to be a useful technique in improving each model's performance. However, they experienced similar drawbacks for each respective model. Namely, the need for further data and computational resources for ViTs are necessary to achive state of the art results. These findings highlight both the promise and the practical limitations of ViT models when applied to small scale datasets without large scale pretraining.

1. Introduction/Background/Motivation

Our project focuses on advancing image classification methods by comparing classical convolutional models with newer models such as transformers and diffusion. Traditional CNNs such as AlexNet and ResNet have driven much of the progress in computer vision, but recent models like the Vision Transformer (ViT) [1] and DiffusionMix [3] have demonstrated new approaches toward these tasks. We explore these approaches hands-on by re-implementing and training them to understand how their design choices affect performance, efficiency, and final accuracy.

This problem is interesting because it highlights a major shift in how vision models are built—from purely convolutional feature extractors to architectures that leverage sequence modeling and probabilistic generation. Understanding these differences is crucial for practitioners who need to choose appropriate architectures for their specific use cases.

For our baseline experiments, we trained a ResNet-32 model on the CIFAR-10 dataset [4] to establish a strong CNN baseline before comparing with transformer and diffusion approaches. CIFAR-10 consists of 60,000 color images of size 32×32 pixels, split into 50,000 training images and 10,000 test images across 10 mutually exclusive classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck). We applied standard data augmentation including random cropping with 4-pixel padding and random horizontal flipping, along with normalization using per-channel mean and standard deviation computed from the training set.

Dosovitskiy et al. [1] introduces the Vision Transformer in their work which had a simple but powerful idea: apply a standard Transformer encoder , the same that was designed for NLP, directly to sequences of image patches. Instead of relying on convolutional networks their ViT design treats each nxn image patch as a token and linearly embeds it, adds positional embeddings, and feeds the resulting sequence into a vanilla Transformer. This approach removes almost all image specific architectural assumptions and tests whether a pure attention based model can learn visual structure directly from data. The Vision Transformer paper evaluates three model sizes, Base, Large, and Huge, which differ mainly in depth, width, and parameter count. ViT-Base (12 layers, 86M params) serves as the lightweight model and performs competitively when trained on large datasets, achieving strong accuracy on CIFAR-10/100 . ViT-Large (24 layers, 307M params) scales capacity and shows substantial improvements The biggest model, ViT-Huge (32

layers, 632M params) delivers the best results across all benchmarks. Overall, they reported that as the models scale from Base to Huge, performance increases consistently demonstrating that simple Transformers, when scaled and pre-trained sufficiently, can surpass convolutional networks.

Islam et al. [1] introduced DiffuseMix which is based on previous data augmentations such as MixUp and Cutmix. Their method leverages the generative priors of diffusion models to create more coherent images. Instead of unnatural distortions, the generative process alters the original image which proved to maintain natural, structural semantics while combining class-discriminative features. Because of their success on CIFAR-100, ImageNet, and data scare regimes, this was the perfect paper to reimplement in order to characterize the benefits of diffusion models for image classification.

In this study we tried to follow a similar approach and train models using this architecture for the same CIFAR10 data set in order to better understand these types of models and also compare the results with the traditional CNN results.

2. Approach

2.1. ResNet

We used ResNet-32 [2], a residual network specifically designed for CIFAR-10, as our baseline CNN architecture. The network consists of an initial 3×3 convolution with 16 filters, followed by three stages of residual blocks with 16, 32, and 64 filters respectively. Each stage contains 5 residual blocks, giving a total of 15 blocks (30 convolutional layers plus the initial convolution and final fully connected layer, hence 32 layers). The architecture concludes with global average pooling followed by a fully connected layer for 10-class classification, totaling approximately 470,000 parameters.

To find optimal training configurations, we conducted a systematic hyperparameter search. We compared SGD with momentum (0.9) against Adam optimizer, explored learning rates of 0.1, 0.01, and 0.001, and tested weight decay values of 5×10^{-4} and 1×10^{-4}. For learning rate scheduling, we used multi-step decay (factor of 0.1 at specified epochs) for SGD and cosine annealing for Adam. All experiments used a batch size of 128 and cross-entropy loss.

We anticipated that learning rate would be the most critical hyperparameter, which proved correct. High learning rates (0.1) worked well with SGD but caused instability with Adam. We also found that the learning rate schedule significantly impacts final performance—without proper scheduling, models either failed to converge or plateaued at suboptimal accuracy. Our implementation used PyTorch [5], with the ResNet-32 architecture implemented from scratch following the original paper specifications.

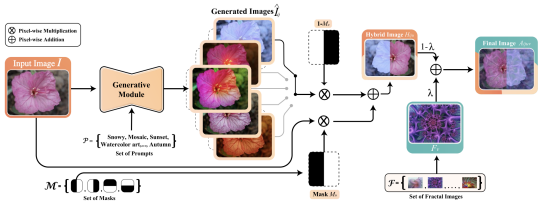


Figure 1: DiffuseMix Architecture

2.2. ViT

As mentioned earlier , the ViT architecture [1] converts an image into a sequence of patch tokens and processes them using a standard Transformer encoder. The image $x \in \mathbb{R}^{H \times W \times C}$ is divided into $N = \frac{HW}{P^2}$ patches of size $P \times P$. Each patch is flattened and projected into a D-dimensional embedding through a learnable linear projection. A learnable class token is prepended, and positional embeddings are added, forming the input sequence:

$$z_0 = [x_{\text{class}};\, x_p E] + E_{\text{pos}}.$$

This sequence is passed through multiple identical Transformer blocks, each consisting of multi-head self-attention (MSA) and a feed-forward MLP, wrapped with LayerNorm and residual connections. Each block updates representations as:

$$z'_\ell = \text{MSA}(\text{LN}(z_{\ell-1})) + z_{\ell-1}, \qquad z_\ell = \text{MLP}(\text{LN}(z'_\ell)) + z'_\ell.$$

After L layers, the final hidden state corresponding to the class token $z_L^{(0)}$ serves as the global image representation, which is then fed into a classification head.

2.3. DiffuseMix

DiffuseMix blends the original image with diffusion generated variations and fractal patterns. These variations enhance robustness and generalization due to semantic and structural perturbations.

Using InstructPix2Pix as the stable diffusion backbone, the prompts were randomly selected from a predefined set (e.g., Snowy, Sunny, Autumn) to guide the transformation coherently. Fractals were generated using the mandelbrot equation:

$$z_{n+1} = z_n^2 + c$$

and then manipulated zoom ($0.5 \times -2.0\times$) and center coordinates. The hopes of this approach was to prevent the model from overfitting to simple textures.

We then combine the Original Image (x_{orig}), the Diffusion-Generated Variation (x_{diff}), and a Fractal Pattern (x_{frac}) using a two-stage mixing process: A binary

mask M is generated, selecting a random vertical or horizontal strip of the image.

$$x_{hybrid} = M \odot x_{orig} + (1 - M) \odot x_{diff}$$

The hybrid image is then linearly interpolated with the fractal image to add structural noise, controlled by a mixing coefficient λ (default $\lambda = 0.15$).

$$x_{final} = (1 - \lambda) \cdot x_{hybrid} + \lambda \cdot x_{frac}$$

For every original image in CIFAR-10, a corresponding DiffuseMix sample is generated. The final training set is the concatenation of the original dataset and these generated samples, effectively doubling the training size.

3. Experiments and Results

3.1. ResNet

We evaluated four hyperparameter configurations, measuring both training and validation accuracy throughout training. Table 1 summarizes our main results.

Configuration	Epochs	Best Acc.
SGD, lr=0.1, wd=1e-4	20	**87.50%**
Adam, lr=0.001, wd=1e-4	10	84.09%
SGD, lr=0.1, wd=5e-4	10	83.72%
SGD, lr=0.01, wd=5e-4	10	82.66%

Table 1: Validation accuracy for different hyperparameter configurations.

Our experiments revealed several important insights about hyperparameter selection for training ResNets. Most notably, SGD with proper learning rate scheduling consistently outperformed Adam across all configurations we tested. The best SGD configuration achieved 87.50% validation accuracy, exceeding the best Adam configuration (84.09%) by 3.4 percentage points. This aligns with prior observations in the literature that SGD with momentum tends to find better minima for image classification tasks, despite Adam's faster initial convergence.

Learning rate emerged as the most critical hyperparameter for SGD. A learning rate of 0.1 significantly outperformed 0.01, with final accuracies of 87.50% versus 82.66% respectively. The multi-step learning rate schedule, which reduces the learning rate by a factor of 10 at epochs 10 and 15, proved essential for continued improvement in the later stages of training. Without this schedule, the model would plateau at suboptimal accuracy. We also observed that lower weight decay (1×10^{-4}) slightly outperformed higher weight decay (5×10^{-4}) when other hyperparameters

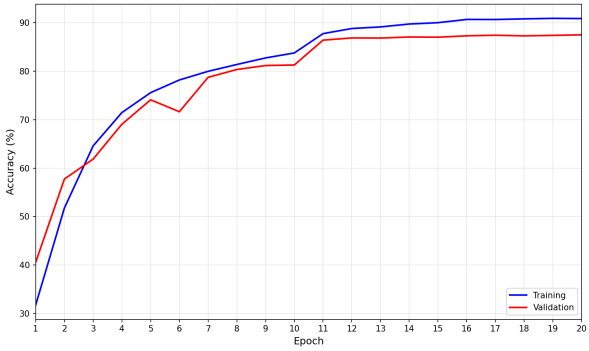


Figure 2: Training and validation accuracy for the best configuration (SGD, lr=0.1, wd=1e-4, 20 epochs).

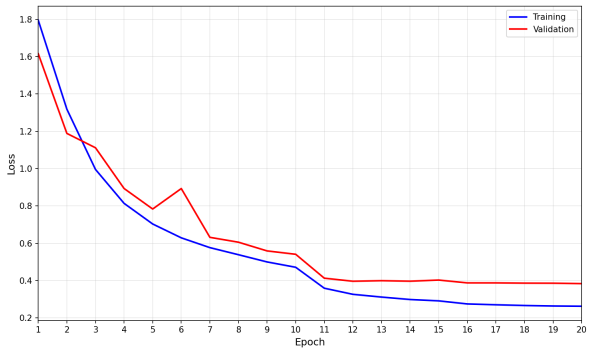


Figure 3: Training and validation loss curves showing convergence behavior.

were held constant, suggesting that excessive regularization can hurt performance on this dataset.

The training dynamics are illustrated in Figures 2 and 3. Figure 2 shows the training and validation accuracy curves for our best configuration, where we can clearly observe the effect of learning rate scheduling. At epochs 10 and 15, when the learning rate is reduced, both training and validation accuracy show notable jumps, demonstrating how learning rate decay helps the optimizer settle into better local minima. Figure 3 shows the corresponding loss curves, which exhibit the expected monotonically decreasing trend with particularly sharp improvements following each learning rate reduction.

We observed a gap between training accuracy (approximately 95%) and validation accuracy (87.50%), which indicates some degree of overfitting. However, this gap is typical for deep networks trained on CIFAR-10, and importantly, the validation accuracy continues to improve throughout all 20 epochs of training. This suggests that the model has not severely overfit and would likely benefit from additional training epochs. Indeed, our best re-

sult of 87.50% is reasonable for ResNet-32 trained for only 20 epochs; the original ResNet paper reports approximately 93% accuracy when training for 200 epochs with similar hyperparameters, indicating substantial room for improvement with extended training.

3.2. ViT

For the ViT architecture, due to computation limitations, in this study 4 smaller models have been trained and tested on CIFAR-10. The models are listed in table 2. Each model then was trained using the Configuration , listed in Appendix I . training was repeated 3 times and the results are listed in table 2.

ViT Models				
	Tiny	**Small**	**Medium**	**Large**
depth	2	4	8	16
patch size	4×4	4×4	4×4	4×4
head	4	4	4	4
epochs	20	20	40	60
params	0.8M	2.1M	6.3M	21.0M
Test Accuracy	$63.33\% \pm 0.05\%$	$68.93\% \pm 0.74\%$	$70.87\% \pm 0.62\%$	$68.03\% \pm 0.82\%$
Test Loss	1.0307 ± 0.0044	1.1204 ± 0.0442	1.8099 ± 0.0308	1.9793 ± 0.0701

Table 2: ViT models params and CIFAR10 test results

3.3. DiffuseMix

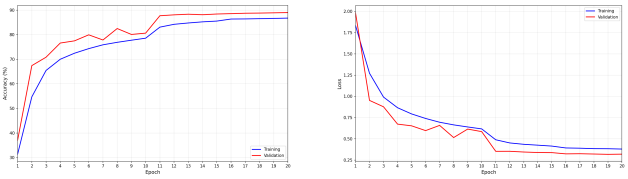
DiffuseMix proved to be a great pipeline for generation and improving the results of the classification models. In Figure 4 we can see the additional images that were generated, bringing the total dataset size from 50k to 100k.

Our evaluation demonstrates that DiffuseMix augmentation consistently alters model performance across varying architectures. As detailed in Table 3, for the ResNet-32 baseline, the augmentation improved the top-performing configuration (SGD, lr=0.1, wd=1 $\times$ 10^{-4}) by 1.49%, achieving a final accuracy of 88.99%. The training dynamics for this configuration are illustrated in Figure 5. The curves exhibit a characteristic step improvement at epoch 11, coinciding with the scheduled learning rate decay. Notably, the validation accuracy consistently tracks above the training accuracy throughout the optimization process.

For Vision Transformers, the impact of the augmentation correlates strongly with model capacity. As shown in Table 3, the Tiny, Small, and Medium variants achieved an average performance increase of over 8%, with the ViT Medium peaking at 79.29%. However, this trend breaks at the upper bound of model size. As illustrated in Figure 9, while the smaller models show healthy convergence over 60 epochs, the ViT Large model (red line) diverges significantly, exhibiting a rapid collapse in validation accuracy early in the training phase and concluding with a performance of 45.92%. Figure 6 presents the individual loss and accuracy curves for all four ViT variants. These plots confirm that while the smaller architectures maintain

Figure 4: DiffuseMix Generated Examples

stable training and validation gaps, the Large model's loss diverges drastically after the initial epochs.



(a) ResNet w/DiffuseMix Accuracy

(b) ResNet w/DiffuseMix Loss

Figure 5: Training and Validation Curves for ResNet w/Diffusion

Configuration	Best Acc.	w/DiffuseMix
SGD, lr=0.1, wd=1e-4	87.50%	**88.99%**
Adam, lr=0.001, wd=1e-4	84.09%	**86.61%**
SGD, lr=0.1, wd=5e-4	83.72%	**87.36%**
SGD, lr=0.01, wd=5e-4	82.66%	**85.16%**
ViT Tiny	63.33 %	**71.58%**
ViT Small	68.93%	**77.08%**
ViT Medium	70.87%	**79.29%**
ViT Large	**68.03%**	45.92%

Table 3: Comparing Results Across all Models w/DiffuseMix

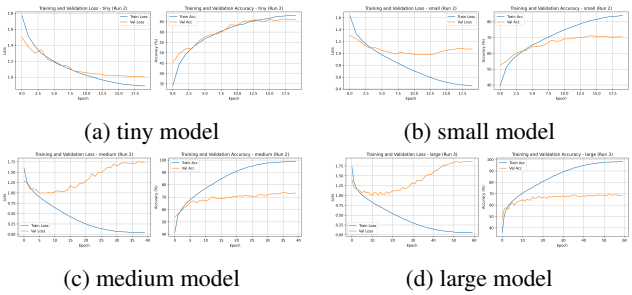
4. Discussion

4.1. ResNet

The success of the ResNet architecture can be attributed to its innovative use of skip connections, which allow gradients to flow directly through the network during backpropagation. This design choice effectively addresses the vanishing gradient problem that plagued earlier deep networks, enabling the training of substantially deeper models. Mathematically, each residual block learns a residual mapping $F(x) = H(x) - x$, where $H(x)$ is the desired underlying mapping. By reformulating the learning objective in this way, the network only needs to learn the deviation from the identity mapping, which is often easier to optimize. This structure is particularly well-suited for image classification tasks where hierarchical features at multiple spatial scales are important for distinguishing between classes.

In our implementation, the learned components include all convolutional filters, batch normalization parameters (scale and shift), and the final fully connected layer weights. These parameters are optimized through backpropagation during training. The fixed components include the network architecture itself, the ReLU activation functions, and the global average pooling operation. The model accepts $32 \times 32 \times 3$ normalized RGB images as input and produces 10-dimensional logits as output. During inference, these logits are converted to class probabilities via the softmax function, and the class with the highest probability is selected as the prediction.

We used cross-entropy loss as our training objective, which is the standard choice for multi-class classification problems. This loss function directly optimizes the log-likelihood of the correct class, providing well-calibrated probability estimates. Our hyperparameter choices were guided by established practices in the deep learning literature, particularly the finding that a learning rate of 0.1 with multi-step decay is a robust recipe for training ResNets on CIFAR-10.



(a) tiny model (b) small model

(c) medium model (d) large model

Figure 6: Training and Validation Curves for ViT Models: (a) Tiny, (b) Small, (c) Medium, (d) Large

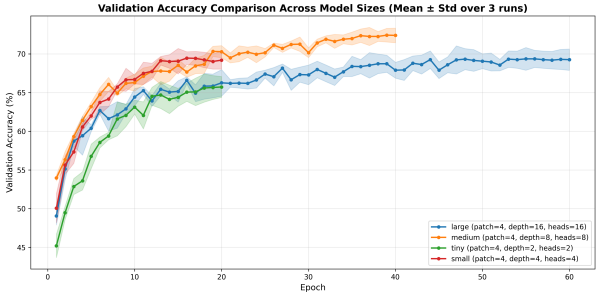


Figure 7: ViT Accuracy vs Epochs for Testing Datasets

4.2. ViT

Looking at the ViT results in Table 2 and Fig **??**, we can observe how model depth affects the architecture's performance on the CIFAR-10 dataset. Four model variants were trained with identical patch sizes (4×4) but varying depth: *tiny* (2 layers), *small* (4 layers), *medium* (8 layers), and *large* (16 layers), all with proportional numbers of attention heads. The medium model achieved the highest performance at $70.87\% \pm 0.62\%$ test accuracy, followed by the small model at $68.93\% \pm 0.74\%$, the large model at $68.03\% \pm 0.82\%$, and the tiny model at $63.33\% \pm 0.05\%$. These results demonstrate the relationship between model capacity and performance, with the largest model underperforming relative to the medium sized variant despite having significantly more parameters and being trained for 60 epochs compared to 20 and 40 for the others.

Looking at the training plots, it can be seen that the *tiny* and *small* models exhibit healthy convergence patterns, with train–validation gaps remaining modest. Their validation losses stabilize or slightly increase toward the end of training, indicating that they are approaching optimal capacity for the dataset. The *medium* model shows similar behavior with slightly larger gaps, but it maintains reasonable generalization performance, with validation accuracy plateauing around epochs 25–30.

In contrast, the *large* model displays severe overfitting: training accuracy reaches 99% while validation accuracy stagnates around 68–71% after epoch 20, creating a train–validation gap of approximately 30%. The validation accuracy comparison plot illustrates these trends clearly. All models show rapid improvement during the first 15 epochs, with the *medium* and *small* models converging fastest to their respective plateaus at roughly 70% validation accuracy. The *large* model's validation curve exhibits high variance and oscillation after epoch 20, with the standard deviation bands widening significantly in later epochs, indicating training instability across runs.

Notably, the *large* model's validation performance peaks early (around epochs 15–20 at approximately 68%) and

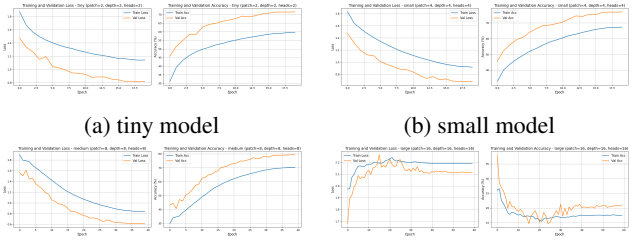
shows no meaningful improvement despite 40 additional training epochs, while training accuracy continues to climb. This divergence between the training and validation curves indicates that the model is overfitting. The *large* model's accuracy of 68%, compared to the *medium* model's 70%, is primarily attributable to severe overfitting caused by excessive model capacity relative to the limited dataset size. With 16 Transformer layers processing only 49,000 training samples, the model is likely memorizing the training data rather than learning generalizable features, as evidenced by the dramatic divergence between training accuracy (99.04%) and validation accuracy (71.20%). Furthermore, the validation loss increases continuously from epoch 20 onward, indicating that additional training beyond early epochs actively harms generalization. Another contributing factor may be suboptimal hyperparameters, which could have caused the *large* model to underperform.

Comparing our results with Dosovitskiy et al. [1] , neither the CNN baseline nor our ViT models reached the accuracy reported in the reference papers. Several factors may explain this discrepancy, including suboptimal tuning and limited computational resources for training deeper or larger models. Nevertheless, even these smaller and suboptimally tuned models perform moderately well, and it can be argued that with better hyperparameter search and greater computational capacity, it would have been possible to reproduce results closer to those reported in the literature. As a reference, the original paper mentioned that their smallest model contains approximately 86M parameters with 12 layers, trained on TPUv3 hardware and using a substantially larger dataset such as JFT-300M for training.

4.3. DiffuseMix

The results indicate that DiffuseMix functions as a potent regularizer, though its limitations lie with the respective model architecture's limitations. For ResNet-32, the inversion of training and validation metrics observed in Figure 5 is particularly telling. This phenomenon—where validation loss drops below training loss—is characteristic of strong data augmentation strategies. It suggests that DiffuseMix generates "hard" training samples that are sufficiently distinct from the source distribution, effectively preventing the model from memorizing training data while enforcing robust feature extraction.

The profound gains observed in the smaller ViT models supports the hypothesis that synthetic diversity introduced by the diffusion process likely force the attention mechanisms to learn more generalized spatial dependencies. Transformers, which lack the inherent translation invariance of CNNs, typically struggle on small datasets like CIFAR-10. The observered scaling limits suggests that while DiffuseMix aids generalization, the largest architectures likely overfitted to synthetic artifacts or suffered from optimization instability due to the increased noise in the training distribution.



(a) tiny model (b) small model

(c) medium model (d) large model

Figure 8: Training and Validation Curves for ViT Models w/DiffuseMix: (a) Tiny, (b) Small, (c) Medium, (d) Large

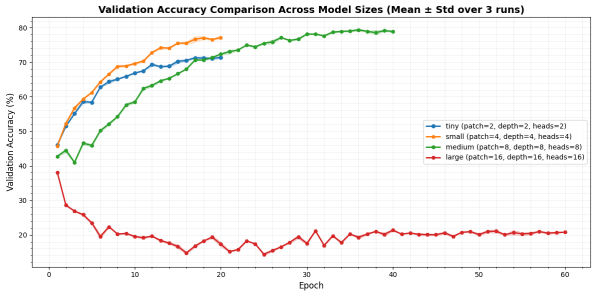


Figure 9: ViT w/DiffuseMix Accuracy vs Epochs for Testing Datasets

4.4. Future Work

Looking ahead, there are several directions for future work. We plan to extend training to 200 or more epochs to approach state-of-the-art accuracy on CIFAR-10. Exploring deeper architectures like ResNet-56 and ResNet-110 would allow us to understand how model capacity affects performance on this dataset. DiffusionMix could see a boost with more robust prompt engineering that is class specific. This would allow data augmentation that is expected in real world scenarios. ViT would see a boost in performance through the extended training. Larger, diverse datasets would also improve ViT as well as improving our overall understanding of these models.

5. Work Division

Table 4 summarizes the contributions of each team member to this project.

References

[1] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner,

Student Name	Contributed Aspects	Details
David Exiga	[Implementation and Experiments]	[Implemented DiffuseMix, trained ResNet-32 and ViT with generated dataset, generated training visualization, and analyzed resutls]
Ali Jalilian	Implementation and Experiments	Implemented ViT architecture, conducted hyperparameter search experiments, generated training visualizations, and analyzed results.
Sara Bahri	Implementation and Experiments	Implemented ResNet-32 architecture, conducted hyperparameter search experiments, generated training visualizations, and analyzed results.

Table 4: Contributions of team members.

Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 1, 2, 6

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 2

[3] Khawar Islam, Muhammad Zaigham Zaheer, Arif Mahmood, and Karthik Nandakumar. Diffusemix: Label-preserving data augmentation with diffusion models. *arXiv preprint arXiv:2405.14881*, 2024. 1

[4] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 1

[5] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019. 2

6. Appendix

A. Source Code

All source code and experiment results are available in the following repository: https://github.com/DALISA-2025/CS7643-final-project.

The following repositories were also used as references for implementing the models: https://github.com/gupta-abhay/pytorch-vit, https://github.com/google-research/vision_transformer, https://github.com/huggingface/diffusers, https://github.com/timothybrooks/instruct-pix2pix and https://github.com/lucidrains/vit-pytorch.

B. ViT Configuration

The following configuration was used for ViT models.

Parameter	Value	Parameter	Value
image_size	32	batch_size	128
patch_size	4	learning_rate	0.0005
num_classes	10	weight_decay	0
dim	256	optimizer	Adam
mlp_dim	512	momentum	0.9
pool	cls	device	cuda
channels	3	dim_head	64
dropout	0.2	emb_dropout	0.1

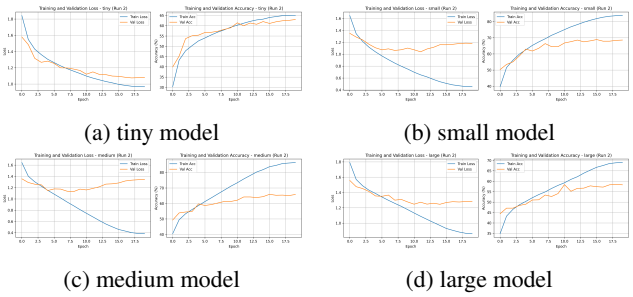
Table 5: Configuration parameters used for the ViT model.

C. ViT extra results

In the first set of experiments, 4 models were trained according to table. the config is similar to the configuration presented in B however the number of epochs was 20 for all the models. Based on the results in 6, it is clear that the bigger models (medium and large) have not been converged yet and therefor the test accuracy are lower. another reason for not seeing the improvement in the larger models is that the 16x16 patch selected for the Large model, will result in 4 patches (2x2 grid) for 32x32 CIFAR10 images, this is extremely coarse-grained and loses fine details and model has very limited spatial information to work with. Therefore a newer set of experiments were conduced, that are summarized in section 3. in there the patch size of 4x4 as a good trade off was selected. also, number of epochs were increased for larger model in order to too a better converge.

Based on the results in Fig 10 the patch size problem is the dominant factor hurting the larger models. The *small* model (patch size $= 4$) achieves the best performance at 70.20% test accuracy, while the *medium* model (patch size $= 8$) drops to 64.67%, and the *large* model (patch size $= 16$) further degrades to approximately 57.8%. This is not a limitation of model capacity; rather, the poor parameterization of the input representation harms the architecture. On 32×32 CIFAR-10 images, a patch size of 16 produces only 4 patches in total, while patch size 8 yields 16 patches, and patch size 4 produces 64 patches. The large model is therefore seeing the image at an extremely coarse resolution with minimal spatial detail, making it unable to learn fine-grained visual features regardless of network depth. Even the *tiny* model, with only two layers but a patch size of 2 (resulting in 256 patches), achieves 59% accuracy competitive with the much deeper *large* model, purely because it uses a richer input representation.

The larger models are also undertrained, showing clear signs of underfitting. From the training curves, the *small* model exhibits validation accuracy plateauing around epochs 15–16, indicating that the model has effectively converged. In contrast, the *medium* and *large* models show validation accuracy continuing to increase steadily even at epoch 20. The *large* model's training accuracy reaches only 69% at epoch 20, suggesting that it has not yet adequately fit the training data.



(a) tiny model (b) small model

(c) medium model (d) large model

Figure 10: Training and Validation Curves for ViT Models: (a) Tiny, (b) Small, (c) Medium, (d) Large

ViT Models				
	Tiny	**Small**	**Medium**	**Large**
depth	2	4	8	16
patch size	2×2	4×4	8×8	16×16
head	2	4	8	16
params	0.8M	2.1M	6.3M	21.0M
Test Accuracy	59.00% ± 0.37%	70.20% ± 0.16%	64.67% ± 1.17%	57.53% ± 0.25%
Test Loss	1.1394 ± 0.0057	1.0885 ± 0.0198	1.2988 ± 0.0253	1.2151 ± 0.0165

Table 6: ViT models params and CIFAR10 test results - extra experiments