CNNs, ViTs, and Stable Diffusion, Oh My!: Exploring Recent Innovations For Image Classification

Daniel Rocha
Georgia Tech
drocha3@gatech.edu

Sidharth Raja
Georgia Tech
sraja34@gatech.edu

Moonish Maredia
Georgia Tech
mmaredia3@gatech.edu

Abstract

Image classification has been one of the most successful applications of Deep Learning. Historically, this success has largely been driven by using CNN architectures and large annotated image datasets. We wanted to explore the continued saliency of these factors given two relatively recent industry innovations: the vision transformer[1] and Stable Diffusion. The vision transformer provided an attention-based alternative architecture to CNNs while Stable Diffusion provided a new approach to generating high-quality images which could serve as an alternative to annotated datasets. In this paper, we leverage both of these innovations. First, we compare, contrast, and analyze the performance and learned characteristics of CNNs vs. Vision Transformers. Second, we evaluate the usefulness of Stable Diffusion as a synthetic data augmentation technique. To explore these issues, we make use of the Stanford Cars Dataset[4], a dataset containing images of vehicles split by make, model, and year with the goal of accurately classifying each vehicle into its appropriate class (make, model, and year).

1. Introduction/Background/Motivation

Image classification is a relevant problem in an uncountable number of settings. Consequently, deep learning models that can accurately classify images can unlock significant commercial and economic value. Historically, these image classification tasks have been dominated by the use of CNNs[1]. However, in 2021, Dosovitskiy et. al released a paper[1] where researchers adapted the Transformer architecture for vision tasks, positing that the architecture was competitive if not better than the state-of-the-art CNN-based Resnet architectures. The results were especially relevant given a Transformer's scaling and computational efficiency making it more trainable than CNNs for very large, complex models. However, as a consequence of the vision transformers' success, it has become less clear whether practitioners should leverage CNN-based architectures or transformer-based architectures and when to use which and how they may differ in terms of performance and what they learn. In a 2022 paper [5], Raghu et. al explored this concept, focusing on comparing the learned features at various layers of the two models via a more technical, metric driven approach. We wanted to extend that work by providing a visual comparison of the two models on a single dataset. Our aim was to evaluate not only relative performance between the models but use common visualization techniques to demonstrate how the learned features differ to help inform what types of problems may or may not be best suited to a certain architecture (with the caveat that our insights will be drawn from a single dataset).

In addition to CNNs, much of image classification has leveraged large-scaled annotated datasets. However, gathering large-scaled annotated data is difficult and expensive. Data augmentation enhances the quality of training dataset, and enables us to train better models. There are many different forms of augmentation [2]. Most commonly, these are image manipulations like geometric (eg: rotation/flip/cropping of training data), photometric (color transforms of training data). For image manipulations, it's easy to use these techniques but they are limited by the amount of training data we already have. In settings where data is scarce, synthetic data can be a potential solution. Based on a 2023 paper by He et al. [7], historic approaches to synthetic data generation have largely relied on GANs. However, GANs are difficult to train and need a lot of initial setup to generate the exact kind of images we need. However, with the advent of powerful text-to-image models such as CLIP and Stable Diffusion, synthetic data generation of high-quality images has become much faster, potentially enabling image classification training without requiring large-scaled annotated datasets. To evaluate how these text-to-image models perform on synthetic data generation tasks, we extend the work done by He et. al [7], which evaluated the use of text-to-image models on common object datasets (e.g. animals, flowers, food). In this paper, we leveraged these generative models to generate very specific data with

highly specific labels to determine whether these text-to-image models can offer meaningful value.

To explore both of these techniques, we leveraged the Stanford Cars Dataset[4] via Kaggle. The dataset contains 16,185 labeled vehicle images, consisting of 8144 images in the train set (50%) and 8041 images in the test set (50%). The images are split across 196 classes where each class represents a unique vehicle make, model, and year (e.g. 2012 Audi A8 Coupe). We explored aforementioned innovations with the goal of accurately classifying vehicles to their respective class. Example images from the dataset are shown in Figure 1

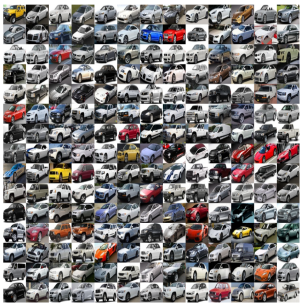

Figure 1. Example images from dataset for 195 classes. Images have been chosen to share common viewpoint

2. Approach

To meet our learning objectives, we undertook several steps. Below, we detail our steps and subsequently present findings in the Experiments and Results section.(**Note**: unless otherwise noted, we used PyTorch library to implement DL models)

2.1. Establishing a Baseline

We started by establishing a performance baseline for the dataset using existing CNN architectures. Specifically, we leveraged 3 common CNN architectures: VGG-16, MobileNetV2, and ResNet-50 without modifications. We tuned and trained these architectures using two mechanisms: 1) randomly initialized weights for all layers and 2) pre-trained ImageNet 1K weights for the convolutional layers and randomly initialized weights for the classifier. Initially, we had hoped to tune all models from purely random initializations but were unable to get meaningful performance given time, resource, and monetary constraints and opted to use pre-trained ImageNet weights to get meaningful performance and analysis. Although we used ImageNet weights upon initialization, weights weren't fixed throughout training (i.e. training updated weights for both the convolutional layers and classifier layers of the CNN). Additionally, given our constraints, we fixed training time for all models (both in this and following sections) to a maximum of 10 epochs to have comparable results throughout. To train the models, we used a 50/50 training and test split. For all models (in this and subsequent sections), standardized pre-processing steps were conducted, which included resizing, random rotation, random horizontal flips, and normalization to make results more robust and generalizable.

2.2. Final CNN and ViT Performance & Performance Comparison

In this section, we took the best model from our baseline CNN architectures and ran additional experiments to see if we could improve the performance via modifications. Given we were using pre-trained weights, we limited ourselves to modifying architecture only at the classifier stage. This included testing additional layers, different regularization techniques, and hyperparameters. We undertook a similar process to determine the best performing Visual Transformer model. To close, we offer a performance comparison of the two models, offering hypotheses and evidence for the differential performance.

2.3. Visualizing CNN and ViT & Comparing Learned Features

In this section, we took our final CNN model and visualized its learned features. We relied on a 3rd party GitHub repository to conduct these visualizations[3]. While we ran several different types of visualization techniques (GradCam, GradCam++, EigenCAM), the reported visualizations leverage EigenCAM. The implementation is based on a paper by Muhammat et. al [6]. Per the researchers, EigenCAM serves as a more general purpose visualization technique for CNNs that relies on PCA to determine learned representations from the convolutional layers, making it independent of class relevance score and backpropagation. The visualizations were built off learned features in the final layer of our ResNet50 model.

We repeated a similar process with the Visual Transformer, utilizing the same 3rd party GitHub repository to conduct these visualizations. For the transformer, we relied on ScoreCAM as it provided the best performance. It worked better than standard GradCAM for ViT as ScoreCam makes use of the attention maps produced by the ViT encoder, addressing a limitation of GradCAM. This visualization algorithm helped us understand where the network was focusing its attention when predicting a certain class, unlike GradCAM, which was building a sparse map focusing on the entire image, mimicking what the encoding layers were doing (encoding the image patches into a sequence to be fed to the network). To close, we leverage CAM visualizations to provide intuitions about what the learned features tell us about the two models.

2.4. Generative Augmentation

Recently, there has been remarkable progress [9] in generative models (specially diffusion models) in generating photo-realistic images based on text and image guidance. These have resulted in the proliferation of open models trained on web scale text and image datasets. Generated samples typically retain enough features such that their content is immediately and correctly recognizable by a human. Thus generative augmentation can be incredibly powerful. In this section, we first describe our process for generating data, and then experimentally explore the following use cases:

- Can generated training examples be used to augment the existing training dataset and help models generalize better, and improve their peak accuracy?

- Can generated examples be used when we don't have many real training examples per class? And in a more extreme setting, if we don't have any training examples for each class?

Note: Generative augmentation in the real world needs to be done with **extreme care** because generated samples perpetuate the biases that went into training the original model.

3. Experiments and Results

3.1. CNN Baseline

Table 5 below summarizes the results of our baseline CNN experiments. All models were trained for a maximum of 10 epochs with an SGD optimizer (momentum = 0.9, regularization=0.0005), CE Loss, and a *ReduceOnPlateau* decay scheduler which decayed learning rate by a factor of 0.25 and patience of 2 epochs. Learning rates and batch sizes were fixed to 0.005 and 64 for all baseline runs. The primary takeaway from the runs was the magnitude of the delta in accuracy between randomly initialized weights and pre-trained weights. While the result was expected, the delta meant that training from scratch was likely infeasible given time and resource constraints. As a result, all training setups in the paper are initialized with pre-trained weights.

3.2. Final CNN and ViT Performance & Performance Comparison

To improve the performance of baseline ResNet50 model, we experimented with a few different modifications to the classifier stage of the model. First, we added an additional FC linear layer to give the model additional expressiveness. However, to prevent overfit, we added regularization mechanisms specifically Dropout and BatchNorm. We also experimented with different learning rates. Post-experimentation, the combination of additional layers, moderate regularization and a slightly higher learning rate resulted in a performance improvement of 4.2% on the test set (83.5% to 87.7%). The combination of the increased complexity coupled with stronger regularization accounted for 1.5-2% of the increase while the increased learning rate accounted for the rest. The final model specifications and tuning details were the following: ResNet50 w/additional linear layer with hidden weights of size (2048, 196), followed by a batch-norm layer, and a dropout operation (w/p=0.3), batch size=64, lr=0.0065, SGD optimizer w/momentum=0.9, CE Loss, and *DecayOnPlateau* lr decay schedule with patience of 2 w/0.25 decay factor. Post-modifications, total model parameters increased from 25.6M to 28.1M. The loss learning curve for the model is presented at the top of Figure 2.

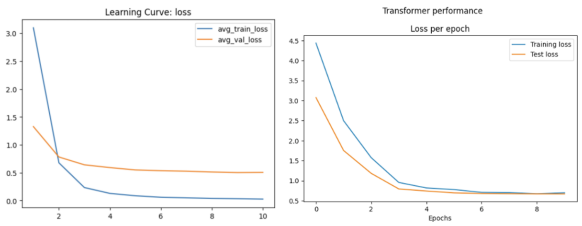

Figure 2. Learning Curve: Loss vs. Epoch. Left: CNN. Rigt: ViT. Note the axes

The Visual Transformer (ViT) architecture chosen was ViT Base 16 [2], with 12 layers and 86M parameters, as implemented by PyTorch's TorchVision module [3]. The model was loaded with ImageNet 1K weights, and the pretrained layers were not frozen - their weights were used as initialization values to speed up convergence. Optimization (*ADAM, weight decay=1e-4, lr=0.0001, stepsize=3, gamma=0.1*) ran for 10 epochs and included all weights, with Cross Entropy Loss used since classes were balanced. The most accurate model across the epochs was saved to disk for further testing.Figure 2 shows the learning curve after 10 epochs. The model did not overfit, but given the plateau reaches around epoch 6, from which there was no more loss reduction, it can be inferred that more training data would be required for better performance. The accuracy of the best model in the test set was 83.8%.

We had initially expected that given the Transformer's success in other domains, it would offer better performance than the ResNet architecture. However, we were surprised to see the ResNet beat the ViT in terms of performance (87.0% vs. 83.8% accuracy). We believe this was due to two main factors: 1) ViT requires larger amounts of data to be trained effectively than CNNs and 2) Transformers do not have the natural inductive bias towards working with images as CNNs do; ViT uses an adaptation of splitting the images into small patches, embedding them linearly, then feeding them to the encoder one by one. In this setup, it may be expected that the relationship between pixels would be better captured by CNN, while the ViT would seek to

Model	Random (Test Accuracy)	Pre-Trained (Test Accuracy)	Number of Parameters
VGG-16	1.0%	70.6%	138,357,544
MobileNetV2	7.0%	80.8%	3,504,872
ResNet-50	2.1%	83.5%	25,557,032

Table 1. Baseline Model Comparison. Random column refers to model trained on dataset using random initialization of weights. Pre-Trained column refers to model trained using pre-trained ImageNet 1K weights

build representations of the labels using a global perspective rather than a small set of localized features.

Further, we hypothesized that the CNN's inductive bias is especially relevant for our dataset. Specifically, the most identifiable aspects of vehicles are the manufacturer's logo and front / back design elements (e.g. headlights, grill, tail lights etc). CNNs are better positioned to learn such local features while ViTs are better position to learn more global features such as the size and overall shape of the vehicle. We tested this hypothesis by identifying the set of cars in which ViT classification performance was highest relative to CNNs and vice versa. The results are summarized in 3 and confirmed our hypothesis. ViTs outperformed CNNs where the shape of the vehicle was a large distinguishing factor (e.g. Cargo Vans and a Hummer SUV). Contrastingly, CNNs outperformed ViTs on sedans which are more similarly shaped and require much more localized information to distinguish. Further, note that although ViTs outperform CNNs on larger vehicles; the absolute CNN performance on those vehicles is still good, which isn't unsuprising because they can still rely on distinct local features, especially for a vehicle like a Hummer. We further reinforce these findings in the upcoming section by visualizing the learned features of each model.

	vehicle_class	cnn_match_ind	vit_match_ind	delta
59	Chevrolet Express Cargo Van 2007	0.172414	0.310345	-0.137931
88	Dodge Dakota Club Cab 2007	0.815789	0.947368	-0.131579
173	Ram C-V Cargo Van Minivan 2012	0.731707	0.853659	-0.121951
0	AM General Hummer SUV 2000	0.909091	1.000000	-0.090909
	vehicle_class	**cnn_match_ind**	**vit_match_ind**	**delta**
17	Audi S4 Sedan 2012	0.641026	0.410256	0.230769
184	Tesla Model S Sedan 2012	0.947368	0.710526	0.236842
180	Suzuki Aerio Sedan 2007	0.842105	0.605263	0.236842
129	Hyundai Accent Sedan 2012	0.833333	0.583333	0.250000

Figure 3. Vehicle Class Comparison. 1st column is vehicle class, 2nd column is CNN accuracy for class, 3rd column is ViT accuracy for class, 4th column is accuracy delta

3.3. Visualzing CNN and ViT & Comparing Learned Features

To visualize what CNNs learned, we utilized EigenCAM. When making high-confidence predictions, CNNs rely heavily on localized features of the vehicles such as headlights, grill, tail lights, wheels, door handles, window shape etc. As Figure 4 shows, when vehicles were classified with certainty, the model had learned specific identifiable localized features of a vehicle. However, on images where distinguishing characteristics were unavailable, the model relied on multiple elements of the car but struggled to classify with certainty.

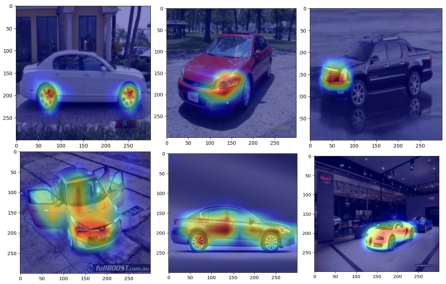

Figure 4. Top image: vehicles with CNN classification with over 95% certainty. Bottom image: vehicles with CNN classification with less than 20% certainty

To understand what was being learned by the ViT, we utilized ScoreCAM. When making high-confidence predictions, ViT does not seem to rely exclusively on singular parts of the car (grille, headlights, etc.) to build its idea of which regions are important. As the top of Figure 5 shows, in this scenario, the model constructs a lower-dimensional encoding of several parts of the car: identifying patterns (probability distributions) over a combination of features, such as: the side door area, windshield, specific curves, overall shape, and some local elements. The bottom of Figure 5 shows the opposite: when the model focuses its attention on smaller image regions, with just one or two parts of the car, it was unable to classify vehicles with high confidence.

These visualizations reinforce the observation that Visual Transformers learn more global and context-dependent features (with relationships between different image patches), versus hard localized features (headlights, wheels, logos). Doing so has the cost that it needs more training data, to be able to establish more relationships, whereas CNNs can perform classification looking a localized features, such as associating the logo with the vehicle make. One such example of the behavior of the ViT can be seen in 6, in which it fails to classify the vehicle but the CNN can classify the vehicle correctly with high confidence. Specifically, the ViT misclassifies a Chevy Crossfire

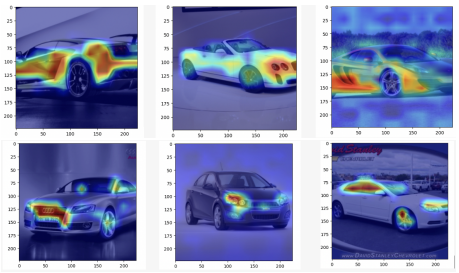

Figure 5. Top image: vehicles with ViT classification with over 95% certainty. Bottom image: vehicles with ViT classification with less than 20% certainty

as a Chevy Sebring because it relies on global vehicle shape and size for classification. However, the two vehicles are similarly sized, with similar shapes, and designs. In fact, the ViT is 100% certain that it's a Sebring. However, the Crossfire has a notable design element near its front wheel. The CNN is able to pick up on feature and seems to rely nearly exclusively on it for classification. The CNN correctly predicts the vehicle as a Sebring with 100% confidence.

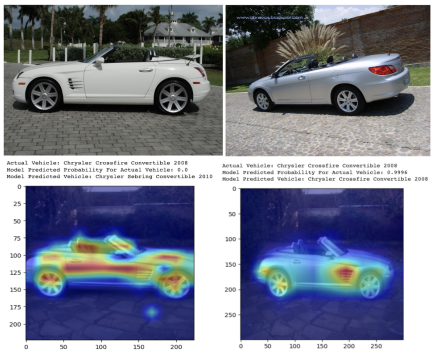

Figure 6. Classification comparison. Top Left: Actual Vehicle (Crossfire) Image From Train Set. Top Right: ViT Predicted Vehicle Image (Sebring) From Train Set. Bottom Left: ViT Learned Features For Crossfire. Bottom Right: CNN Learned Features For CrossFire

The results provide some meta takeaways about the strengths and weaknesses of the two models. First, while the Transformer has largely replaced older methods (such as RNNs, LSTMs) for NLP tasks, ViTs aren't obviously superior. Because images contain localized features that can help distinguish them, the CNN is likely well-suited for tasks in which images have highly localized distinguishing factors. However, we expect CNNs will struggle when these localized features aren't available in high-quality. For instance, most of our vehicle images were high quality and well-illuminated. In real-world settings (e.g. traffic monitoring), where images are lower quality and localized features are less likely to be visible (such as logo, lights), we expect Transformers are much more robust. (We attempted to test this hypothesis but were unable to find traffic images of similar classes). This means that although Transformers haven't unseated CNNs altogether, depending on the image types and model application, they may be much more robust.

3.4. Generative Augmentation

We used Stable Diffusion v1.5 [8] to generate our training examples. We used the txt2img model without image guidance. We described desirable features using the prompt, and the undesirable features using the negative prompt. The negative prompt is crucial to avoid distortions and disfigured features in images.

prompt: a photo of a (class) car on a (scenario), full-body, ultrarealistic

car_class draws from 196 classes in Stanford dataset

scenario draws from a predefined list of environments (eg: desert, highway, mountain, forest, country)

negative prompt: ugly, bad, uncool, distorted, obscene, disfigured

Figure 7. Generated images for Porsche Panamera (above) and Smart fortwo (below) in different environments

We generated 50 samples for each class (so 9800 new images) based on different versions of the prompt mentioned above. We split this into 80% train and 20% val images. We discovered that some generations were flagged for NSFW. These were removed without replacement.

We have uploaded our generated and augmented dataset here (Generated Cars Dataset) and the combined training dataset here (Augmented40 Cars Dataset) for review / use.

3.5. Improving peak accuracy

We setup an experiment to determine whether training on (generated + real) images can result in better model test accuracy compared to training on just the real images. We wanted to find out if this augmentation could help any model and evaluated this using two different architectures.

- Model A = A Resnet50 with the final layer changed to match the number of output classes. It was pre-trained on Imagenet1k and fine tuned with all layers (batchSize=32, epochs=10, lossFn=CrossEntropy, optimizer=Adam, LR=0.001, no LRscheduler)

- Model B = This is the optimized ResNet50 CNN architecture described earlier (all parameters and details are the

same but model was trained on real + augmented data)

For both Model-A and Model-B, training with (real + generated) training images yielded a higher test accuracy compared to training just with the real images. (Figure 8) This means training with generated augmentation can be a useful way to squeeze additional performance from existing models.

	Base	Augmented
Model-A	80.1%	82.0% (+1.9pp)
Model-B	87.7%	88.8% (+1.1pp)

Table 2. "Base" uses only real training examples; "Augmented" uses real training examples + 40 generated examples per class

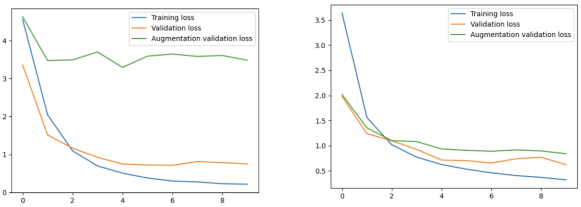

Figure 8. Model-A Loss curves for training with real + 0 gen per class (left); and real + 40 gen per class (right)

3.6. Augmenting when there's less real data

Training mix per class	Test Accuracy
3 real, 0 augmented	13.30%
3 real, 3 augmented	16.20%
3 real, 5 augmented	18.84%
3 real, 10 augmented	25.92%
3 real, 15 augmented	28.31%
3 real, 20 augmented	31.56%
3 real, 35 augmented	29.14%
3 real, 40 augmented	34.16%

Table 3. (Model-A) trained with different data mixes

In this experiment, we restricted the number of real images in each class (to 3), and varied the amount of generated images in the training mix. As we increased the amount of augmentation, we hoped to see an increase in the test accuracy. This indicates that our model is able to learn features of each vehicle class from the generated data and that those features are transferable to the real test data. Our hypothesis was that this would start diminishing (and may even be counterproducitive) after a point if we kept increasing the mix of generated data because it would overfit to specific patterns of the generated data. In our testing thus far, (at +40 generated per class), we hadn't noticed diminishing returns. We could theoretically keep increasing the mix of generated data to find that limit, but generating entire datasets via diffusion is a costly / resource intensive process.

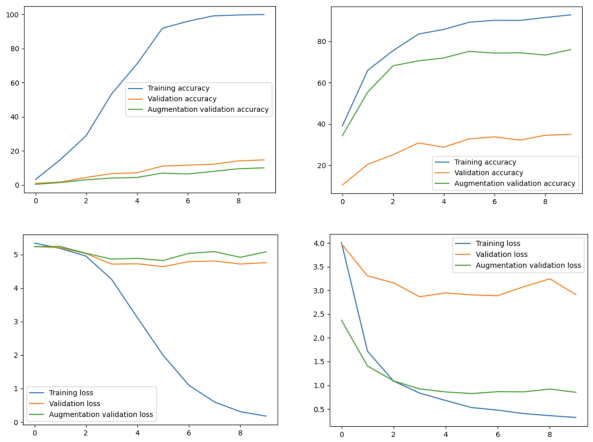

Figure 9. Accuracy and loss curves: (Left: 3 real + 0 generated) and (Right: 3 real + 40 generated) as training mix per class

We can see that that for 0 augmentations, val accuracy (real data) increases but augmentation_val accuracy (generated data) is still pretty low. However for 40 augmentations per class, the augmentation accuracy is higher than the validation accuracy which reflects the training mix. The key takeaway is that even with limited real data, leveraging generated data can result in generalizable accuracy on the real data. However, the generated data isn't a total replacement for real data. The test accuracy is not nearly as high as when we trained with the full real data.

3.7. Training on only generated data

Lastly, we explored the scenario when there's no real data available, so we trained on only the generated images (40 per class). The performance was quite poor (test accuracy = 25.35%). However, interestingly some classes performed relatively well with only generated data. We believe this was due to the relative uniqueness of these vehicles (i.e. difficult to mistake for other cars). This means that generated data can be a useful augmenting technique in classes with sparse data and relatively unique characteristics.

Car Class	Class Accuracy
smart fortwo Convertible 2012	82.5%
Rolls-Royce Ghost Sedan 2012	78.9%
Lamborghini Reventon Coupe 2008	77.8%

Table 4. Prediction accuracy for 3 highest performing classes

Underperformance while training on only generated samples is likely because having only generated images causes the model to overfit to generated examples and features. However as shown in Sec 3.6, retaining even a small amount of real training examples likely keeps the model grounded in reality and helps it learn common attributes from real and generated datasets and help it generalize better.

4. Work Division

See Table 5.

References

[1] Alexander Kolesnikov Dirk Weissenborn Xiaohua Zhai Thomas Unterthiner Mostafa Dehghani Matthias Minderer Georg Heigold Sylvain Gelly Jakob Uszkoreit Neil Houlsby Alexey Dosovitskiy, Lucas Beyer. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021. 1

[2] Taghi M. Khoshgoftaar Connor Shorten. A survey on image data augmentation for deep learning. https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0197-0, 2019. 1

[3] Jacob Gildenblat and contributors. Pytorch library for cam methods. https://github.com/jacobgil/pytorch-grad-cam, 2021. 2

[4] Michael Stark Jonathan Krause, Jia Deng and Li Fei-Fei. Collecting a large-scale dataset of fine-grained cars. *CVPR 2015*, 2015. 1, 2

[5] Simon Kornblith Chiyuan Zhang Alexey Dosovitskiy Maithra Raghu, Thomas Unterthiner. Do vision transformers see like convolutional neural networks? 2022. 1

[6] Mohammed Yeasin Mohammed Bany Muhammad. Eigen-cam: Class activation map using principal components. 2020. 2

[7] Xin Yu Chuhui Xue Wenqing Zhang Philip Torr Song Bai Xiaojuan Qi Ruifei He, Shuyang Sun. Is synthetic data from generative models ready for image recognition? *ILCR 2023*, 2023. 1

[8] RunwayML. Stable diffusion 1.5. https://huggingface.co/runwayml/stable-diffusion-v1-5, 2022. 5

[9] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Yingxia Shao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *arXiv preprint arXiv:2209.00796*, 2022. 3

5. Data References

Dataset can be found on Kaggle.

We are also including links to Generated Cars Dataset and Augmented40 Cars Dataset

Student Name	Contributed Aspects	Details
Daniel Rocha	ViT Training, Tuning, Visualization, and Comparative Analysis	Trained and tuned the final Transformer model, experimented with and generated ViT visualizations, performed comparative analysis on CNN / ViT performance and learned features with Moonish Maredia.
Moonish Maredia	CNN Training, Tuning, Visualization, and Comparative Analysis	Trained and tuned models for CNN baselines, trained and tuned the final ResNet50 model, experimented with and generated CNN visualizations, performed comparative analysis on CNN / ViT performance and learned features with Daniel Rocha, implemented a simple occlusion scheme to generate an occluded dataset to test if CNNs can learn like ViTs (*implementation details and results excluded for paper due to page limits*)
Sidharth Raja	Generative Augmentation, Experimentation, and Analysis	Responsible for generative augmentation: creating and cleaning the dataset, designing experiments that could showcase the effect of different mixes of augmentation, training multiple models on multiple runs to highlight its contributions. Worked on CNN baseline for Mohsin car dataset (*we ultimately decided not to use that dataset*).

Table 5. Contributions of team members.