Style Transfer and Improved CNN Performance Using GAN-Based Data Augmentation

Abdul Basit
Univeristy of Evry Paris Saclay
Paris, France
basitmal36@gmail.com

***Abstract*—This paper focuses on improving CNN classifiers using GAN-based data augmentation. There are many applications where we have limited data, and the performance of CNN models is not good in these cases. To improve the CNN performance, we have done data augmentation using GAN and presented the results that are evidence of improved performance.**

Keywords—Generative Adversarial Network (GAN), Conditional Generative Adversarial Network (C-GAN), Convolutional Neural Network (CNN)

I. INTRODUCTION

Convolutional Neural Networks (CNNs) have achieved remarkable success in image classification tasks. However, their performance is highly dependent on the quantity and diversity of the training data. In many practical scenarios, collecting large, well-labeled datasets is expensive, time-consuming, or simply infeasible. This limitation often results in suboptimal performance of CNN models when trained on small datasets. To address this challenge, our project investigates the use of Generative Adversarial Networks (GANs) for data augmentation to enhance CNN classifier performance. The project is structured into four main tasks. **Task 1:** We begin by training multiple CNN classifiers on a limited dataset. These include a custom CNN model, ResNet-50, VGG16, and VGG19 architectures. Both randomly initialized and pre-trained variants of ResNet-50, VGG16, and VGG19 are evaluated using 4-fold cross-validation to establish a baseline. **Task 2:** We implement GAN-based data augmentation. Starting with a basic GAN, we progressively improve the model by using soft labels and then employing Conditional GANs (C-GANs). We further enhance the quality of generated data through progressive growing of the generator and discriminator architectures, moving from 64×64 to 128×128 and ultimately 256×256 resolution images. The best quality synthetic images are used for augmentation. **Task 3:** We explore style transfer using CycleGAN to demonstrate domain translation capabilities and enhance dataset diversity by transferring styles between domains while preserving content. **Task 4:** We repeat the classification experiments from Task 1 using the original dataset combined with the synthetic data generated in Task 2. The results show that GAN-augmented data significantly improves the performance of the CNN models, particularly in the pre-trained ResNet-50 and VGG architectures.

II. CNN CLASSIFIERS

To systematically evaluate the impact of GAN-based data augmentation on classification performance, we trained a variety of CNN architectures on a limited dataset under two main configurations: randomly initialized models trained from scratch and pre-trained models fine-tuned on our dataset. The training and evaluation were performed using 4-fold cross-validation to ensure robustness and reduce overfitting bias.

A. Randomly Initialized Models from Scratch

In this configuration, the CNN architectures were initialized with random weights and trained solely on our original dataset. The models included:

Custom CNN: A lightweight convolutional architecture designed specifically for our dataset, composed of 4 convolutional layers followed by ReLU activations, max pooling layers, and fully connected layers for classification.

ResNet-50: A deep residual network with 50 layers that incorporates identity skip connections to mitigate the vanishing gradient problem.

VGG16 and VGG19: Deep convolutional models with 16 and 19 layers, respectively, using sequential stacks of 3×3 convolutional layers followed by max pooling.

All models were trained for a fixed number of epochs(50) with standard optimization techniques (e.g., Adam optimizer), cross-entropy loss, and learning rate scheduling. Performance metrics were averaged across the four folds to obtain consistent results.

B. Pre-Trained Models

To evaluate the benefits of transfer learning, we also fine-tuned pre-trained models initialized with weights from ImageNet. These models included **pre-trained ResNet50, VGG16** and **VGG19.**

For each model, the final classification layer was replaced with a custom fully connected layer matching the number of target classes in our dataset. During training, earlier convolutional layers were optionally frozen or unfrozen depending on validation performance. The same 4-fold cross-validation scheme was applied to maintain comparability with models trained from scratch.

C. Cross-Validation (Cross Training and Testing)

The dataset was split into 4 folds using a 50-50 training-testing scheme in each fold. In every iteration, 50% of the data was used for training and the remaining 50% for testing, ensuring that all data samples were eventually used for both training and evaluation.

III. Data Augmentation

A. Generative Adversarial Network (GAN)

To introduce diversity in the training set, a **Generative Adversarial Network (GAN)** was utilized for synthetic image generation. GANs are a class of unsupervised learning models introduced by Goodfellow et al., consisting of two neural networks: a **Generator (G)** and a **Discriminator (D)**, trained in an adversarial manner. The Generator learns to map a latent vector sampled from a noise distribution (typically Gaussian) to realistic images, while the Discriminator learns to distinguish between real and generated (fake) images. The training objective is a minimax game where G aims to maximize the probability of D misclassifying fake images, while D aims to minimize classification error.

In this project, both networks were implemented using convolutional neural networks (CNNs) in PyTorch. The Generator is an encoder-decoder architecture that upsamples the latent vector of size 100 through multiple transposed convolutional layers with batch normalization and ReLU activations, producing 256×256 RGB images. The Discriminator is a deep CNN that down samples the input image through a series of convolutional layers with LeakyReLU activations and batch normalization, producing a scalar probability via a sigmoid activation function.

The GAN was trained on the real training dataset using the **Binary Cross-Entropy (BCE)** loss function. The Adam optimizer was used for both networks with a learning rate of 0.0002 and β values of (0.5, 0.999), which are known to stabilize GAN training. To track training dynamics, generator and discriminator losses were recorded across 500 epochs. Fixed noise vectors were used to visualize consistent outputs across epochs, allowing for qualitative assessment of generator improvements. Every 50 epochs, generated samples were visualized and inspected.

After training the GAN with hard labels (0 for fake, 1 for real), we also experimented with **soft labels** (e.g., 0.9 for real, 0.1 for fake) to stabilize training and reduce overconfidence in the discriminator, which led to smoother convergence and improved image quality.

B. Conditional Generative Adversarial Network (C-GAN)

The Conditional Generative Adversarial Network (cGAN) builds upon the standard GAN architecture by introducing label conditioning, enabling controlled image generation based on class-specific information. In a typical GAN, the generator creates images solely from random noise, and the discriminator learns to distinguish real from fake images. However, in cGANs, both the generator and the discriminator are provided with additional input—class labels—to guide the generation and evaluation process.

In our implementation, the generator receives a concatenation of the random noise vector (z) and a one-hot encoded class label vector. This combination ensures that the generator learns to associate specific noise patterns with corresponding classes, allowing it to produce images that align with the given labels (e.g., generating a horse when conditioned on the "horse" class). On the discriminator side, the input image is paired with the same class label vector, which is embedded and processed alongside image features to assess whether the image is both realistic and class-consistent.

This modification significantly improves the control over the generation process, allowing for **class-aware image** item synthesis or cross-domain image translation.

C. Progressive Growing

After implementing the **Conditional GAN (cGAN)**, we further enhanced the model by incorporating **progressive growing** of the image resolution. Progressive growing is a technique that gradually increases the resolution of the generated images during training, starting from smaller dimensions and progressively reaching the target size. This approach stabilizes training and improves the quality of generated images by allowing both the generator and discriminator to learn low-level features at smaller resolutions before tackling more complex high-level features at larger resolutions.

In our case, the resolution starts at **64x64** pixels and progressively increases through **128x128** to **256x256** pixels. At each stage, additional convolutional layers are added to the generator and discriminator to handle the increased complexity. The generator first learns to produce coarse features at the low resolution and gradually refines the image as the resolution increases. Similarly, the discriminator adjusts to evaluate finer details as the resolution grows.

This progressive training method has been shown to significantly improve image quality and model stability. By starting with simpler, low-resolution images and gradually introducing more details, the model avoids issues such as mode collapse and enables the generator to focus on the structure and content of the image before adding intricate details like textures and fine-grained features. The final result is more realistic and coherent high-resolution images, with the generator effectively capturing both the global structure and local details at **256x256** resolution.

D. Data-Augmented CNN Models

After applying data augmentation to the original dataset, we generated fake images using a CycleGAN. We then selected a subset of these fake images, amounting to 50% of the original dataset size, and added them to the training set to improve class diversity and robustness.

To evaluate the performance, we trained a ResNet-50 classifier using 4-fold cross-validation. The model was trained in two configurations: i) Randomly initialized weights, ii) Pretrained on ImageNet. This allowed us to compare the performance gains from both data augmentation via GAN and transfer learning.

IV. Style Trasnfer

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save

As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper; use the scroll down window on the left of the MS Word Formatting toolbar.

A. Cycle GAN

Style transfer refers to the task of transforming the style of an image while maintaining the content. For instance, given a photograph, style transfer can alter its appearance to resemble that of a painting or an artistic rendering. CycleGAN (Cycle-Consistent Generative Adversarial Network) offers a robust method for **unpaired image-to-image translation**, which allows style transfer without needing paired training datasets. This makes CycleGAN a powerful tool for tasks like artistic style transfer, where the goal is to apply a specific visual style (e.g., an artistic painting style) to a source image while preserving its content.

CycleGAN consists of two generators and two discriminators, each operating on different image domains:

Generator GABG_{AB}: Transforms images from domain A (e.g., horse images) to domain B (e.g., cow images).

Generator GBAG_{BA}: Transforms images from domain B to domain A.

Discriminator DAD_A: Evaluates how realistic images from domain A are.

Discriminator DBD_B: Evaluates how realistic images from domain B are.

The CycleGAN architecture leverages adversarial training and **cycle consistency loss**. The adversarial loss encourages the generators to create realistic images, while the cycle consistency loss ensures that when an image is translated from one domain to the other and then back, the original content is preserved. This dual loss mechanism helps preserve the **content** of the image while applying the desired **style**.

***Generator**: The generator in CycleGAN is a **ResNet-based architecture** with residual blocks. Each residual block consists of convolutional layers followed by Instance Normalization and ReLU activations. The generator's objective is to map an image from the source domain to the target domain while preserving content. The generator uses a **U-Net architecture**, which is helpful for preserving spatial information across transformations.*

*a) **Discriminator**: The discriminator is a **PatchGAN** model, which classifies whether each patch in an image is real or fake. This model focuses on local image patches rather than the global image to help evaluate the quality of the generated images more effectively.*

b) CycleGAN uses the following loss components:

*c) **Adversarial Loss**: The adversarial loss is computed for both domains. The generator GABG_{AB} is trained to fool DBD_B, and GBAG_{BA} is trained to fool DAD_A.*

*d) **Cycle Consistency Loss**: The cycle consistency loss ensures that the original image can be recovered after it has passed through both generators. This loss helps preserve the image content during the transformation.*

The objective function of CycleGAN combines the adversarial and cycle consistency losses.

V. Results

A. Randomly initialized CNN Models

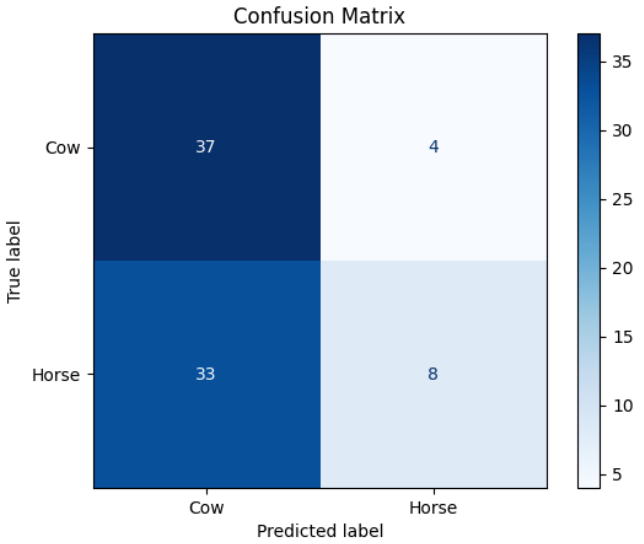

Figure 1: This shows the confusion matrix of the Custom CNN model with an accuracy of 54.88%

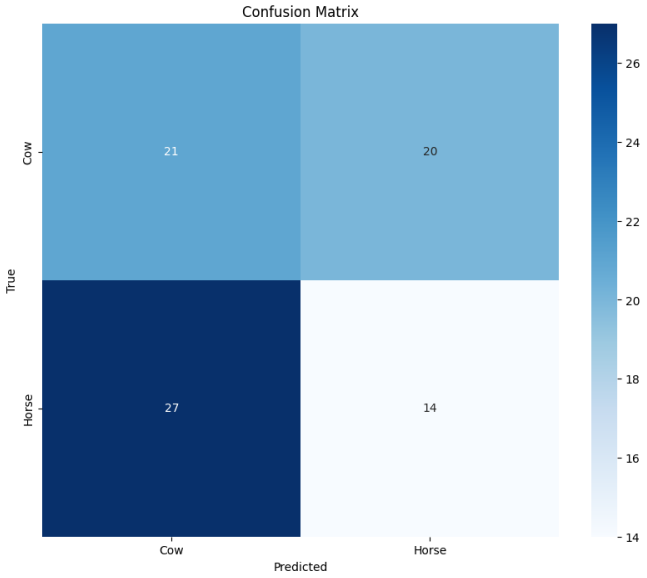

Figure 2: This shows the confusion matrix of the VGG16 CNN model with an accuracy of 42.68%

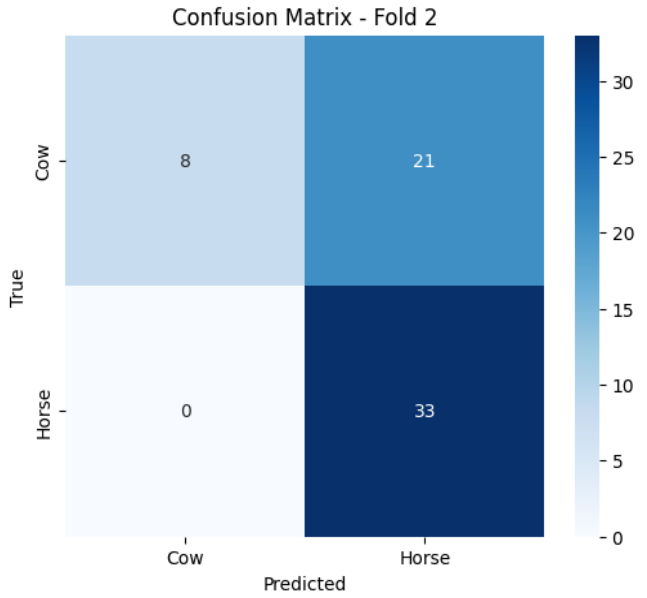

Figure 3: This shows the confusion matrix of the ResNet50 CNN model with an accuracy of 66.13% obtained after 4-fold training and testing.

Fold 1 Testing Accuracy	66.13%
Fold 2 Testing Accuracy	66.13%
Fold 3 Testing Accuracy	62.90%
Fold 4 Testing Accuracy	61.29%

This table shows the testing accuracies at respective folds.

B. Pre-trained CNN Models

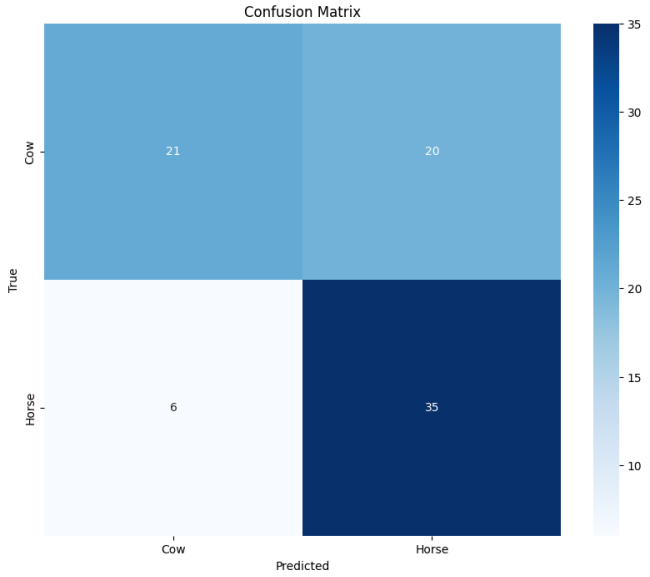

Figure 4: This shows the confusion matrix of the pre-trained VGG19 CNN model with an accuracy of 68.72%

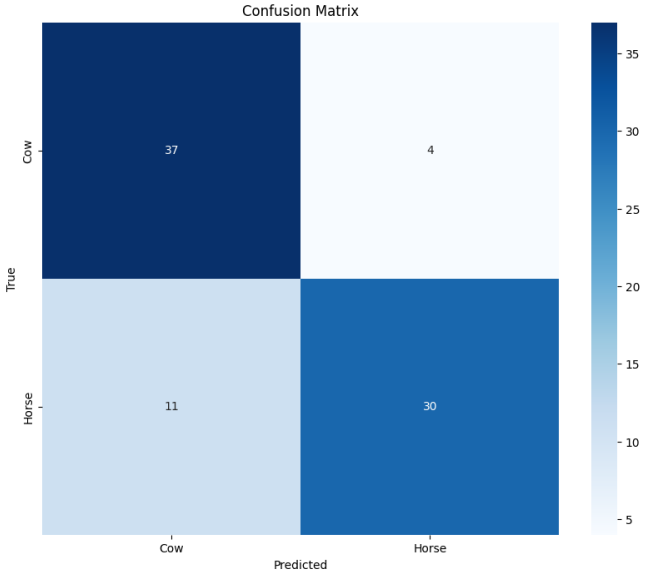

Figure 5: This shows the confusion matrix of the pre-trained VGG16 CNN model with an accuracy of 81.72%

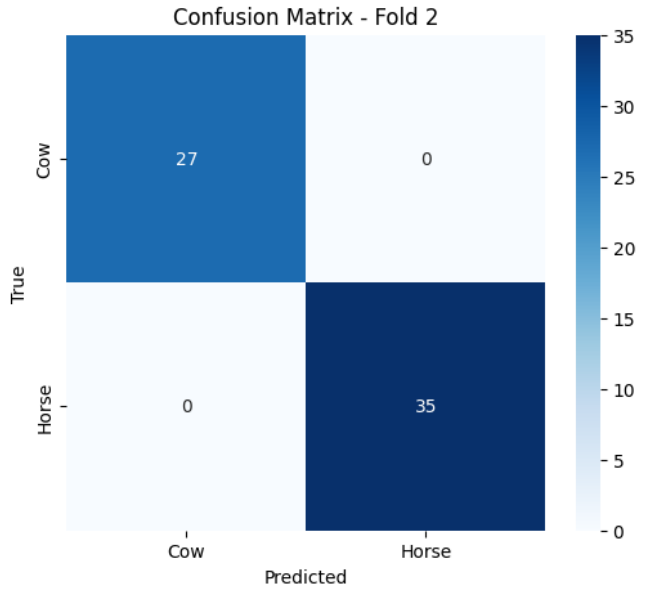

Figure 6: This shows the confusion matrix of the ResNet50 CNN model with an accuracy of 100 % obtained after 4-fold training and testing.

Fold 1 Testing Accuracy	96.77%
Fold 2 Testing Accuracy	100%
Fold 3 Testing Accuracy	98.39%
Fold 4 Testing Accuracy	98.39%

This table shows the testing accuracies at respective folds.

C. GAN

Figure 7: This shows the generated images using simple GAN model.

D. GAN with soft labels

Figure 8: This shows the better generated images using simple GAN model with soft labels.

E. Conditional GAN

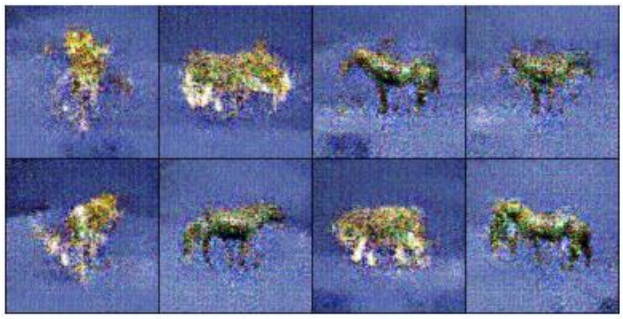

Figure 9: This shows the generated images using simple conditional GAN model.

F. Conditional GAN with progressive growing

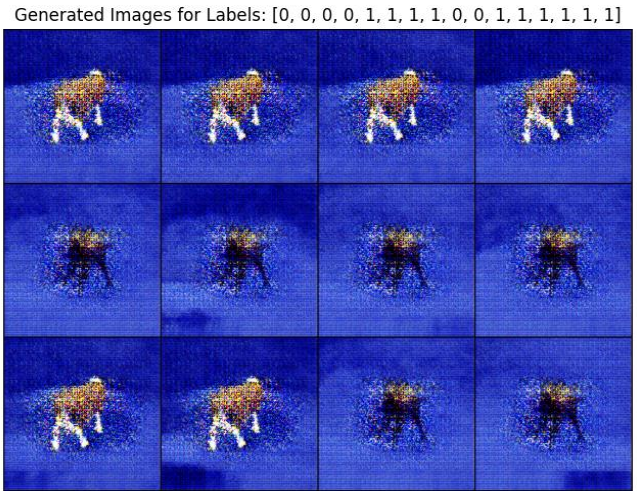

Figure 10: This shows the better generated images using simple conditional GAN model with progressive growing.

G. Augmented CNN Models

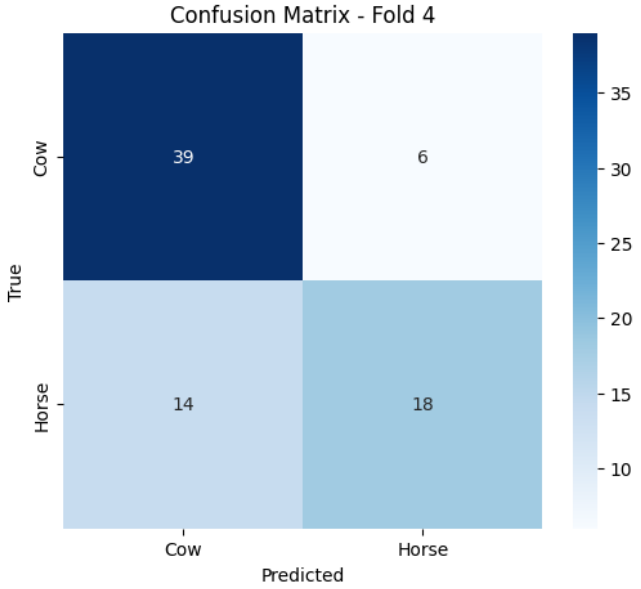

Figure 11: This shows the confusion matrix of the ResNet50 CNN model with an accuracy of 74.033% obtained after 4-fold training and testing.

Fold 1 Testing Accuracy	63.64 %
Fold 2 Testing Accuracy	55.84%
Fold 3 Testing Accuracy	59.74%
Fold 4 Testing Accuracy	74.03%

This table shows the testing accuracies at respective folds.

H. Cycle GAN

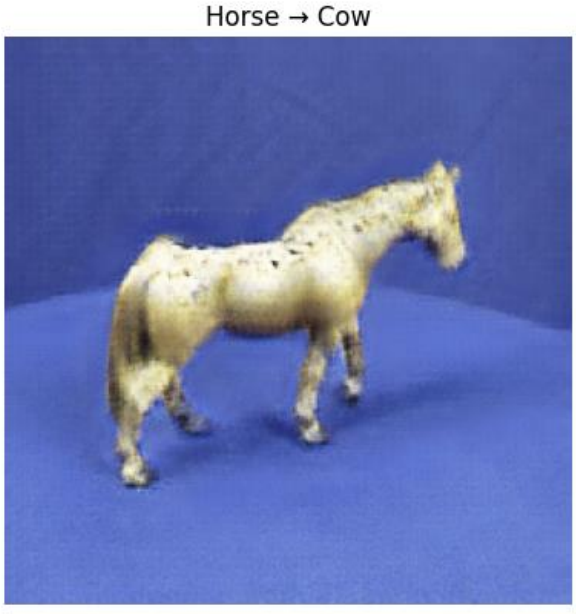

Figure 12: This shows the translated image from horse to cow using unpaired image translation

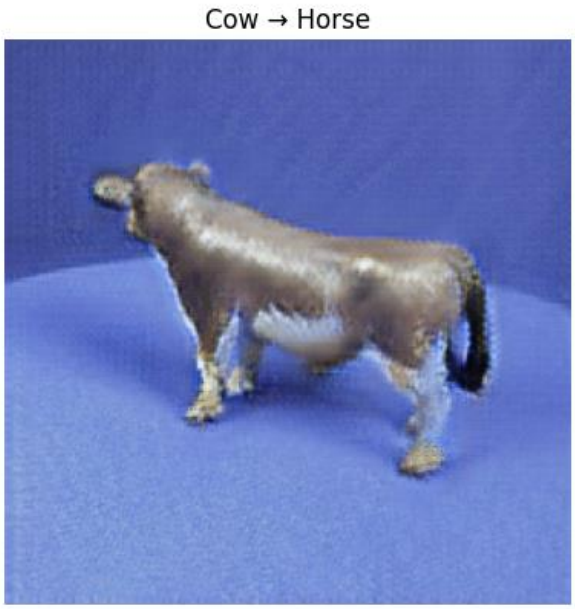

Figure 13: This shows the translated image from cow to horse using unpaired image translation

VI. Conclusion

In conclusion, we are successfully able to improve the performance of the CNN model by adding augmented data generated by a conditional GAN using progressive growing, in which our accuracy for randomly initialized ResNet50 has increased from 66.01% to 74%. I was also able to transfer the style of two classes, respectively, using CycleGAN.

References

[1] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2015. [Online]. Available: https://arxiv.org/abs/1409.1556

[3] I. Goodfellow *et al.*, "Generative adversarial nets," *Advances in Neural Information Processing Systems*, vol. 27, 2014. [Online]. Available: https://arxiv.org/abs/1406.2661

[4] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014. [Online]. Available: https://arxiv.org/abs/1411.1784

[5] J.-Y. Zhu, T. Park, P. Isola and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," *2017 IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, 2017, pp. 2223–2232, doi: 10.1109/ICCV.2017.244.