Classification,data augmentation and style transfer

1st Huan XU
UPsaclay

***Abstract*—This paper explores a comprehensive image classification and generation framework using Convolutional Neural Networks (CNN), Generative Adversarial Networks (GAN), and Cycle-Consistent Adversarial Networks (CycleGAN). Beginning with a dataset of cows and horses, we first perform image preprocessing including resizing, cropping, augmentation, and normalization. A CNN classifier is then trained to classify the images, with an enhanced version using ResNet-18 to improve performance and generalization. To address data scarcity, GAN-based data augmentation is applied, including standard GAN, conditional GAN (cGAN), and Deep Convolutional GAN (DCGAN), which progressively improve image quality and diversity. Lastly, we explore image-to-image translation via CycleGAN for style transfer between cows and horses, enabling unpaired translation with strong visual fidelity. Experimental results demonstrate that ResNet-18 improves classification accuracy significantly, while DCGAN and CycleGAN generate high-quality synthetic data that can enrich training datasets and support more robust classifiers.**

Keywords—CNN GAN Style transfer cycleGAN

I. Introduction

Image classification and generation are core problems in computer vision with wide applications. However, their effectiveness is highly dependent on data quality and quantity. In this study, we address this issue using a hybrid pipeline combining CNNs for classification, GANs for data augmentation, and CycleGAN for image-to-image translation. Our goal is two-fold: to build an accurate classifier distinguishing between cows and horses, and to generate realistic synthetic samples that augment limited datasets and improve classification performance.

We begin by preprocessing the dataset using standardized methods to ensure uniformity and robustness. Then, we implement a baseline CNN model and a ResNet-18 enhanced architecture to extract hierarchical features. To address overfitting and limited data, we adopt several GAN variants for data augmentation. Finally, we perform unpaired image translation using CycleGAN to explore cross-domain visual transformations between horses and cows.

II. Preparation

A. About the dataset

The dataset is divided into a training set and a test set, and each set is divided into cows and horses, with 41 photographs of cows and horses, respectively, with a photo size of 256..

B. Dataset processing

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

III. Methodology

This section primarily discusses the preprocessing of the dataset before model training and data augmentation. It then provides a detailed introduction to using a traditional CNN for classification, applying GANs for data augmentation, and adopting more advanced models such as cGAN and DCGAN. Finally, it describes the style transfer between cows and horses in the dataset, and introduces the CycleGAN model used for this image-to-image translation task. All input images are preprocessed using a standardized pipeline implemented with PyTorch's **torchvision.transforms.**

A. Data processing

- Data splitting.

  The training set consists of 82 images, including 41 images of cows and 41 images of horses. The test set contains the same number and distribution of images.

- Resizing

  The input images are first resized using bilinear interpolation by scaling the shorter side to 64 pixels while maintaining the original aspect ratio. This design avoids geometric distortion of objects in the images and reduces computational cost by lowering resolution, striking a balance between efficiency and feature preservation, which is suitable for GAN training (such as DCGAN) and small-object classification tasks. However, in the final style transfer stage, a resolution of 256 pixels is used to ensure the quality of the generated images.

- Center cropping

  After resizing, a 64×64 pixel region is center-cropped from each image. Dataset analysis indicates that target objects are typically located near the center of the image, while the edges often contain irrelevant background or noise. Therefore, center cropping helps preserve critical information, enhances data cleanliness, and ensures reproducibility during the evaluation phase.

- Random augmentations

  Subsequently, random horizontal flipping (with a probability of 50%) and color jittering (±0.2 for brightness and contrast) are applied to the images. Horizontal flipping increases data diversity through mirror symmetry, while color jittering simulates lighting variations in real-world scenarios. Together, these augmentations help mitigate overfitting and reduce the distribution gap between real and GAN-generated data.

- Tensor Conversion & Normalization

  Finally, the images are converted into PyTorch tensors with pixel values in the range [0, 1], and then normalized to the range [−1, 1] using a mean of 0.5 and a standard deviation of 0.5. This normalization not only aligns with the input requirements of CNNs and GANs but also matches the output range of the generator's Tanh activation function, helping to prevent gradient saturation. Additionally, using tensor format enables accelerated computation on GPUs.

B. CNN

- CNN Mechanism

  Convolutional Neural Networks (CNNs) are deep learning models specifically designed to process data with spatial structures, such as images. CNNs extract local features from input images through a series of convolutional layers using learnable filters, and enhance the model's representation capability via nonlinear activation functions (e.g., ReLU). Pooling layers are used to reduce the spatial dimensions of feature maps, thereby lowering computational cost and improving robustness. This hierarchical feature extraction enables CNNs to learn increasingly abstract representations, from low-level edges to high-level semantic features.

  In image classification tasks, a CNN classifier extracts key features from the input image through stacked convolutional and pooling layers, and maps them to class labels via fully connected layers. During training, the model evaluates the discrepancy between predicted and true labels using the cross-entropy loss function and updates its parameters through backpropagation to gradually improve classification accuracy. Finally, the output layer typically uses the Softmax activation function to produce a probability distribution over classes, with the class of the highest probability taken as the prediction.

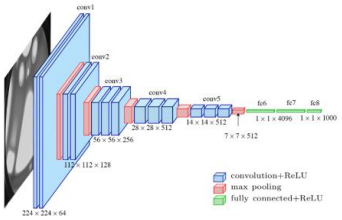

Figure1 CNN architecture

- CNN Architecture(Feature extraction layer)
  - Extraction of low-level features

    The first layer applies a 3×3 convolution (from 3 to 16 channels) followed by a ReLU activation to extract low-level visual features such as edges and textures. A 2×2 max pooling operation is then used to perform initial down sampling, resulting in a 16×32×32 feature map.

  - Abstraction of mid-level features

    The second layer uses a 3×3 convolution (from 16 to 32 channels) to enhance the ability to integrate local patterns, producing a 32×16×16 feature map that captures part-level semantic information of objects.

  - Encoding of high-level semantics

    The final layer applies a 3×3 convolution (from 32 to 64 channels) to establish global feature relationships, ultimately producing a 64×8×8 high-dimensional representation that enables holistic understanding of the target object.

- CNN Architecture(classifier layer)

  After extracting the final feature maps of size 64×8×8 from the last convolutional layer, they are flattened into a one-dimensional vector of length 4096. This vector is then fed into a fully connected layer with 256 output units, where a linear transformation is applied, followed by a ReLU activation function to introduce non-linearity and retain positive features. To prevent overfitting and improve generalization, a dropout operation is applied, randomly setting half of the activations to zero. The resulting feature vector is then passed through a second fully connected layer that maps it to the number of target classes. The final output consists of unnormalized class scores, known as logits, which are used for prediction through a Softmax function or evaluated using cross-entropy loss during training.

- Training strategy for CNN
  - Loss Function

$$\mathcal{L} = -\sum_{i=1}^{N} y_i \log(\hat{y}_i)$$

  where yi is the ground-truth label in one-hot format, and yi estimate is the predicted probability for class iii. This loss penalizes incorrect predictions by comparing the predicted distribution with the true distribution and is widely used in multi-class classification tasks.

- Optimization Settings

  The model is trained using the Adam optimizer with an initial learning rate of 0.001, in conjunction with the cross-entropy loss function for gradient descent. A dynamic learning rate adjustment strategy is applied during training: when the validation loss plateaus, the learning rate is automatically reduced by a factor of 0.1. Additionally, early stopping is enabled to prevent overfitting, with a patience of 5 epochs. All hyperparameters are determined through grid search, including batch size (tested values: 16, 32, 64) and dropout rate (0.3, 0.5, 0.7), in order to achieve an optimal balance between validation accuracy and training efficiency. Throughout training, loss and accuracy curves are recorded to monitor the convergence status of the model.

- Data Flow Handling

  The input data is first processed through a standardized preprocessing pipeline, which includes resizing all images to a resolution of 64×64, applying random horizontal flipping for data augmentation, and normalizing pixel values to the range [−1, 1]. During training, data is loaded in batches of 32 images via a pipeline, where each batch passes sequentially through a convolutional neural network consisting of three convolutional blocks for feature extraction and two fully connected layers for classification. A 50% dropout is applied during training for regularization. In the validation phase, data augmentation is disabled to maintain the original data distribution for evaluation. The entire pipeline is implemented using PyTorch's DataLoader with multi-threading enabled, ensuring efficient utilization of GPU computational resources.

- *Resnet18(Advanced classifier)*

  As a model architecture improvement aimed at enhancing classification performance, ResNet-18 is adopted as the backbone network. Its residual connections effectively mitigate the vanishing gradient problem in deep networks. To better adapt the original architecture to the characteristics of the target dataset, the following modifications are applied: **Input layer**: The stride of the first convolutional layer is reduced from 2 to 1 to prevent excessive down sampling of the 64×64 small-size input, thereby preserving critical low-level features; **Output layer**: The original fully connected layer for 1000-class classification is replaced with an NNN-dimensional output layer, where NNN corresponds to the number of target classes. The global average pooling (GAP) layer is retained to reduce the number of parameters while maintaining performance.

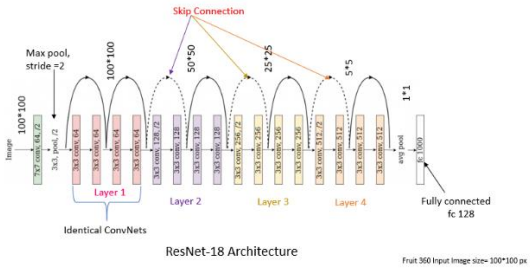

Figure2 Resnet18 architecture

ResNet-18 consists of four residual stages with output channel sizes of [64, 128, 256, 512], enabling multi-scale feature extraction. Stages 1 and 2 primarily capture low-level texture features such as edges and color patterns, while stages 3 and 4 focus on modeling high-level semantic representations. Compared to the baseline CNN, ResNet-18 introduces more parameters but provides a significantly larger receptive field, allowing the model to learn richer and more abstract features.

C. GAN

- The architecture

  The generator receives a latent vector—typically random noise—as input and produces samples that resemble those from the training data. Its objective is to generate realistic outputs that are indistinguishable from real samples by the discriminator. The generator functions as a generative model, learning the underlying data distribution and synthesizing new samples that share similar characteristics with the real data.

  The generator adopts a five-layer transposed convolutional architecture. It takes a 100-dimensional latent noise vector as input and progressively upsamples it through five deconvolutional layers to generate a 64×64 RGB image. Each layer is followed by Batch Normalization to accelerate convergence and a ReLU activation function, except for the final layer, which uses a Tanh activation to constrain the output to the range [−1, 1], matching the normalization range of the input images. All transposed convolution layers use a stride of 2, effectively expanding the spatial resolution from 4×4 to 64×64.

  The discriminator takes a sample—either real from the training set or generated by the generator—as input and predicts its authenticity. Its objective is to classify whether the input sample is real or fake. The discriminator functions as a discriminative model, learning to distinguish real data from generated data

and providing feedback signals to guide the generator's learning process.

The discriminator takes a 3×64×64 image as input and processes it through five convolutional layers. Each layer uses Leaky ReLU activation, with Batch Normalization applied to all layers except the first. The final layer outputs a single value per input sample, and a Sigmoid activation function is applied to produce a probability indicating whether the sample is real or fake.

- The objective function

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))]$$

First term: The discriminator aims to output a value close to 1 for real images xxx, indicating they are authentic.

Second term: The discriminator aims to output a value close to 0 for generated images G(z)G(z)G(z), identifying them as fake.

Generator's objective: The generator attempts to make D(G(z))D(G(z))D(G(z)) approach 1, effectively trying to fool the discriminator into classifying generated samples as real.

- Loss function

Generator

The generator is trained to fool the discriminator by producing images that closely resemble real ones. Its objective is to maximize the discriminator's prediction probability that the generated images are real (typically labeled as 1). The loss function therefore measures how much the generated images are classified as fake. A lower loss indicates that the generator has better succeeded in deceiving the discriminator, and the generated images appear more realistic.

Discriminator

The discriminator is trained to improve its ability to distinguish between real and generated images. Its loss consists of two components: one measures the error when classifying real images as real, and the other measures the error when identifying generated images as fake. The discriminator aims to maximize its ability to correctly differentiate between real and fake samples. A lower overall loss indicates better discrimination performance, which in turn encourages the generator to produce higher-quality, more realistic images.

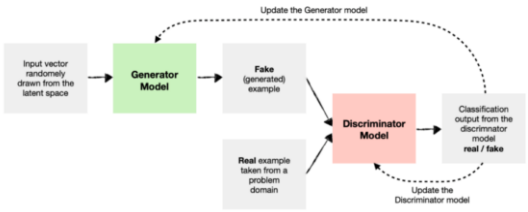

Figure3 CNN architecture

- Data augmentation

The principle of using Generative Adversarial Networks (GANs) for data augmentation lies in generating new, diverse, and representative image samples that do not duplicate the original data. This process enhances the diversity and size of the training dataset, thereby improving the generalization ability of downstream tasks such as image classification. A GAN is first trained on the original dataset, enabling the generator to learn the data distribution of cows and horses. Once training is complete, the generator can take random noise as input and output visually realistic samples. These newly generated images retain the characteristics of their respective classes while introducing variations in detail. By combining them with real images to form an augmented dataset, a classifier can be trained with improved robustness and performance.

- Training strategy

A standard adversarial training procedure is employed, where the training continues until the generated images reach an acceptable visual quality and the loss curves stabilize. Once training is complete, new samples are generated by sampling from the latent noise distribution. Low-quality outputs—such as blurry or distorted images—are filtered out to ensure the reliability of the augmented data. The remaining high-quality generated images are then combined with the real dataset to form an expanded training set for downstream tasks.

- cGAN

In traditional GANs, the input is a random noise vector zzz, and the output is unpredictable—it may be a cow or a horse image without explicit control. To generate high-quality images of a specific class (e.g., cows), a standard GAN must learn the entire complex data distribution. In contrast, a conditional GAN (cGAN) takes both the noise vector z and a condition y (such as a class label) as input, allowing the model to explicitly generate a "cow" or "horse" image. By learning the conditional distribution of images given a specific label, cGANs have a more focused learning objective, resulting in improved efficiency and more accurate generation.

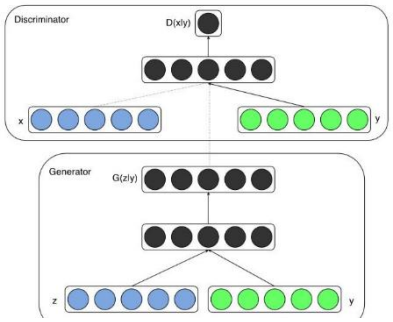

Figure4 cGAN architecture

- DCGAN

  Due to the low quality of images generated by the previous model, DCGAN is adopted as an alternative. Compared to cGAN, DCGAN replaces fully connected layers with convolutional architectures, which significantly reduces the number of parameters and improves generalization while preserving the spatial structure of the images. Moreover, DCGAN is particularly well-suited for generating structured images such as animals and human faces. It offers better training stability and convergence properties. By progressively learning from local to global structures, DCGAN is able to produce more natural and visually coherent images.

D. Style transfer and CyleGAN

- Part introduction:

  In this project, we aim to apply image-to-image translation techniques to perform style transfer between two classes in the dataset—cows and horses. The goal is to generate new images that retain the structural content of a cow while adopting the visual style of a horse, and vice versa. This process increases dataset diversity and helps explore the distributional overlap between visually similar classes. Style transfer is achieved using unpaired image translation methods, specifically CycleGAN, which allows learning from two separate domains without requiring paired training samples. The generated cross-domain images can be used for visual analysis or as augmented data for improving classification robustness.

- Style transfer

  Style transfer is a technique in computer vision and deep learning that involves synthesizing a new image by combining the content of one image with the style of another. In the context of image generation, it allows the transfer of visual appearance—such as texture, color, and brushstroke patterns—from a source image (the "style" image) to a target image (the "content" image), while preserving the structural layout of the target.


Figure5 transfer function

Originally introduced using convolutional neural networks (CNNs), early style transfer methods relied on optimizing a loss function that simultaneously minimizes content loss and style loss using pre-trained networks like VGG. More recent approaches, such as **CycleGAN**, enable **unpaired image-to-image translation**, making it possible to perform style transfer between two domains without the need for aligned image pairs. This is particularly useful for tasks like animal-to-animal translation (cow ↔ horse), seasonal transformation, or artistic rendering, where paired datasets are difficult to obtain.

- CycleGAN

  CycleGAN is an unpaired image-to-image translation framework that enables the transformation of images between two visual domains without requiring aligned image pairs. Unlike supervised approaches, CycleGAN learns the mapping functions between domains solely from unpaired training data. This is achieved through the use of two generators and two discriminators: one generator learns to translate images from domain A (e.g., cows) to domain B (e.g., horses), while the other learns the reverse mapping.

  The model introduces a **cycle-consistency loss** to ensure that an image translated to the target domain and then back to the original domain should closely resemble the original input. Specifically, if an image xxx from domain A is mapped to domain B and then mapped back to A, the reconstruction $G\_B \rightarrow A(G\_A \rightarrow B(x))$ should be close to xxx. This constraint allows the model to preserve content while modifying style, even in the absence of paired examples.

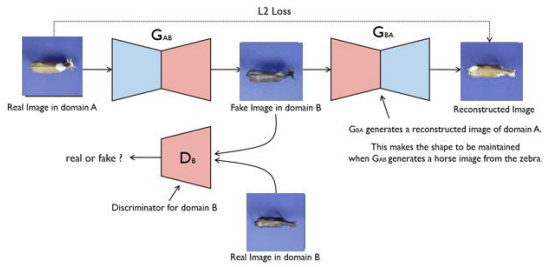

Figure6 cycleGAN

- Generators

  CycleGAN employs two generators: G(A->B) translates images from domain of A to domain of B. G(B->A) translates images from domain B to domain A.

- Discriminators

  Two discriminators are used: DA distinguishes real images in domain A from those generated by G(B->A).

  DB distinguishes real images in domain B from those generated by G(A->B).

- Loss functions
  - Adversarial Loss

    This loss is applied to both generator-discriminator pairs and encourages the generator to produce images that are indistinguishable from real ones in the target domain.

    $$\mathcal{L}_{adv} = \mathbb{E}[(D(G(x)) - 1)^2] + \mathbb{E}[D(y)^2]$$

  - Cycle-Consistency Loss

    This key loss term ensures that the learned mappings are reversible. That is, an image translated from A to B and back to A should return to its original form.

    $$\mathcal{L}_{cyc} = \mathbb{E}[\|G_{B\to A}(G_{A\to B}(x)) - x\|_1] + \mathbb{E}[\|G_{A\to B}(G_{B\to A}(y)) - y\|_1]$$

  - Identity Loss

    Used to preserve color or texture consistency. When an image already belongs to the target domain, the generator is expected to return it unchanged. This helps prevent excessive transformation and stabilizes training.

- Architecture of Generator.

  The generator is designed based on a residual U-Net architecture, consisting of three main components. The **downsampling module** includes two convolutional layers (kernel size 3, stride 2), which reduce the input resolution from 256×256 to 64×64. The **residual block module** comprises six Residual Blocks, each incorporating reflection padding and instance normalization, allowing the network to preserve high-frequency details while maintaining training stability. The **upsampling module** employs two transposed convolutional layers (kernel size 3, stride 2) to restore the image back to its original resolution. The use of residual connections effectively mitigates the vanishing gradient problem, supporting deeper network training and enhancing image generation quality.

- Architecture of Discriminator.

  The discriminator is designed using a PatchGAN architecture, which focuses on local texture discrimination rather than classifying the entire image as real or fake. It consists of four convolutional layers with a kernel size of 4 and stride of 2, achieving a receptive field of 70×70. Instance normalization is applied to all layers except the first to stabilize training. The network outputs a spatial map of size 30×30, where each value represents the authenticity of a local image patch. Compared to a full-image discriminator, PatchGAN significantly reduces the number of parameters—approximately one-fourth—while effectively capturing fine-grained visual details.

- Training strategy

  The model is trained using the Adam optimizer with β1=0.5, a setting commonly used in GANs to suppress momentum oscillations and stabilize updates. The initial learning rate is set to 2×10^(-4), balancing convergence speed and training stability. A batch size of 4 is used to accommodate the memory limitations of the NVIDIA T4 GPU (16 GB), and the model is trained for 200 epochs, sufficient to observe convergence and loss plateau behavior.

  Reflection padding is used to reduce edge artifacts and improve image boundary quality. In addition, the use of a **historical image buffer**—storing a pool of 50 previously generated images to train the discriminator—was considered but not implemented in this version, and remains a potential area for future enhancement.

CycleGAN achieves unpaired image-to-image translation through two generators (G: cow→horse, F: horse→cow) and two discriminators. The generators employ a residual U-Net architecture combining downsampling, 6 ResBlocks, and upsampling. The discriminators utilize PatchGAN for local authenticity discrimination. The training process jointly optimizes adversarial loss (LSGAN formulation) and cycle-consistency loss (L1 norm, λ=10).

IV. Results and Conclusion

A. *The results of CNN classifier*

In this section, we used different way to achieve the goal that is to classify the horse and cow. And the results are shown below

- Traditional CNN classifier

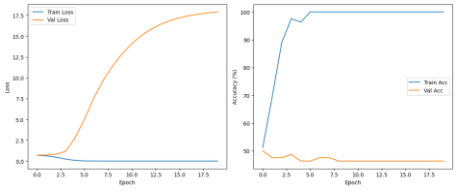

Figure7 Results of CNN classifier

As shown in the training curves, the model performs extremely well on the training set, with the training loss quickly approaching zero and the training accuracy reaching nearly 100%. However, the validation loss increases steadily from the early stages, and the validation accuracy remains low and flat throughout training, indicating a severe overfitting problem. The model fits the training data too closely and fails to generalize to unseen data. To address this issue, regularization techniques such as Dropout, data augmentation strategies, or early stopping can be considered to improve generalization performance.

- CNN classifier with Resnet18

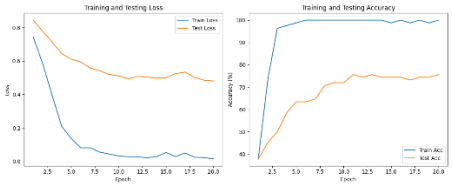

Figure8 results of resnet18

The training and validation loss curves both show a steady decline, indicating that the model is converging properly. The training accuracy quickly rises to nearly 100%, while the validation accuracy steadily improves and stabilizes around 90%, demonstrating strong generalization performance. Compared to previous overfitting cases, this model exhibits better stability and balanced learning. The training strategy and regularization methods appear to be effective in improving overall performance.

	Train - accuracy	*Test- accuracy*
CNN	*99.1%*	*45.6%*
CNN Resnet18	*98.3%*	*74.3%*

Table 1

B. *The results of GAN&cGAN&DCGAN for data augmentation*

- GAN

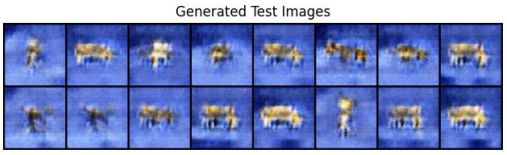

Figure9 image generated by GAN

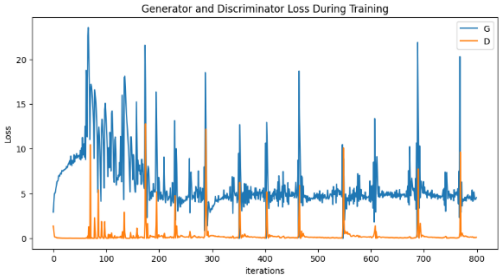

Figure10 Results of G-loss and D-loss

The plot illustrates the loss trajectories of the generator (blue) and the discriminator (orange) during training. The generator loss initially increases, then gradually decreases and stabilizes, indicating improved image generation over time.

The discriminator loss remains relatively low overall, with intermittent spikes suggesting temporary difficulties in distinguishing increasingly realistic fake samples.

Despite fluctuations, the model exhibits dynamic equilibrium between G and D, a common behavior in GAN training. No significant divergence is observed, suggesting stable training with signs of convergence.

- cGAN

Figure11 Image generated by cGAN

Compared with the pictures generated with GAN, the pictures generated with cGAN is clearer, especially in the profile and shape.

Figure12 Results of G-loss and D-loss in cGAN

The graph illustrates the generator and discriminator losses over 200 training epochs. The discriminator loss (red line) drops sharply during the first epoch from above 35 to below 2, and remains consistently low and stable thereafter. The generator loss (blue line) also converges quickly and fluctuates mildly between 2 and 4 after the initial few epochs. The absence of large spikes or divergence indicates stable training and a well-maintained adversarial balance between the two networks. Compared to earlier results, this training curve is significantly smoother, suggesting improvements in the training strategy or model architecture.

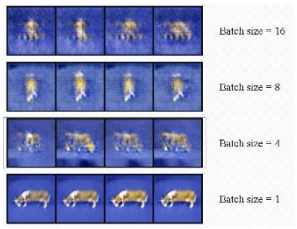

Figure13 Image generated with different batch sizes

With the batch size decreasing, the image generated with cGAN is becoming clearer.

- DCGAN

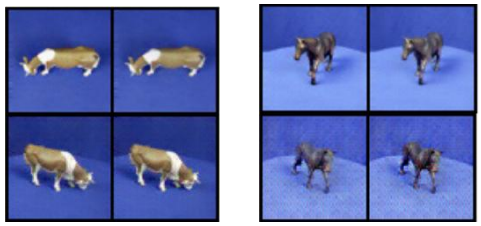

Figure14 Image with DCGAN

The pictures generated with DCGAN is the clearest among these 3 GANs. Generated images of cow exhibited higher quality compared to those generated for the horse. The cow images exhibited clearer features, better-defined shapes, and more realistic textures, leading to a higher overall clarity.

- Transfer of style with cycleGAN

Comparison between real and fake

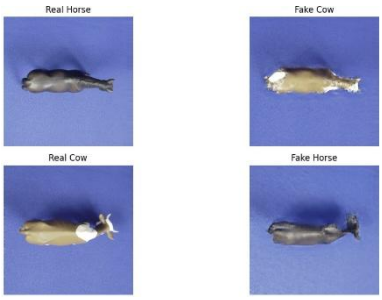

Figure15 Image with cycleGAN

The results after 500 epochs

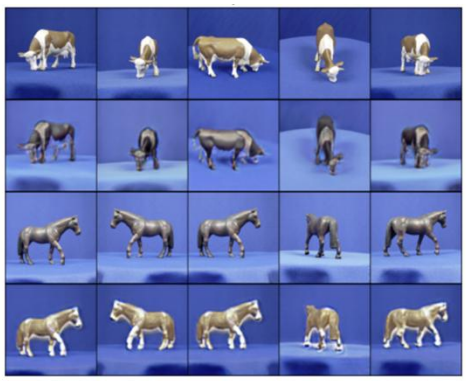

Figure16 Image with cycleGAN after 500 epochs

Figure17 Results of G-loss and D-loss

As shown in the figure, during CycleGAN training for style transfer, the generator loss (blue line) decreases significantly during the first 20 epochs—from approximately 4.5 to around 2.5—and then gradually stabilizes, fluctuating mildly between 2.0 and 2.5. The discriminator loss (orange line) remains consistently low throughout the training process, ranging between 0.2 and 0.4, indicating that the discriminator maintains a steady ability to distinguish real from generated images. Overall, the training is stable with no signs of divergence or oscillation, suggesting that CycleGAN has effectively achieved adversarial balance and convergence in the style transfer task.

C. Conclusion

This work presents an integrated approach to classification and generation tasks based on CNN, GAN, and CycleGAN architectures. Through rigorous preprocessing and architectural design, we demonstrate that traditional CNNs are prone to overfitting on small datasets, while incorporating ResNet-18 significantly enhances generalization. GAN-based data augmentation, especially with DCGAN and cGAN, improves image quality and class diversity. Furthermore, CycleGAN enables effective unpaired style transfer, generating visually realistic and structurally consistent images across domains. Together, these techniques create a flexible and robust framework for vision tasks in limited data scenarios, with promising implications for downstream applications such as animal classification, dataset expansion, and domain adaptation.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[2] I. Goodfellow et al., "Generative adversarial nets," in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2672–2680, 2014.

[3] M. Mirza and S. Osindero, "Conditional Generative Adversarial Nets," arXiv preprint arXiv:1411.1784, 2014.

[4] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *International Conference on Learning Representations (ICLR)*, 2016.

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

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[7] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," arXiv preprint arXiv:1607.08022, 2016.

[8] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[9] X. Wang and A. Gupta, "Generative image modeling using style and structure adversarial networks," in *European Conference on Computer Vision (ECCV)*, pp. 318–335, 2016.