Deep Learning: Pets Facial Expression Classification and Style Transfer

Group 3: Fuqian Zou, Glenys Lion, Iris Lee, Liana Bergman-Turnbull

Northwestern

Agenda

1. **Problem Statement & Overview**
2. **EDA**
3. **Image Classification**
   - 3.1 Custom CNN Model
   - 3.2 Transfer Learning
4. **Style Transfer**
   - 4.1 Neural Style Transfer
   - 4.2 Fast Style Transfer
   - 4.3 CycleGAN
5. **Conclusion & Future Work**
6. **Appendix**

Problem Statement & Overview

Image Classification

Objective:
Classify pet facial expressions into happy, sad, angry and other

Motivation:
- Helps pet owners to understand the pets well-being by understanding the pets emotional state
- Potential use in AI-assisted pet monitoring system

Model Approaches:
- Image preprocessing and augmentation
- Baseline Custom CNN
- Transfer Learning (ResNet, MobileNet)
- Training Strategies (Optimizer, Early Stopping, Learning Rate Scheduling)

Model Evaluation:
- Training and Validation Learning Curves (Model accuracy and loss)
- Test Performance Metrics: Accuracy, Precision, Recall , F1-score, Confusion Matrix

Style Transfer

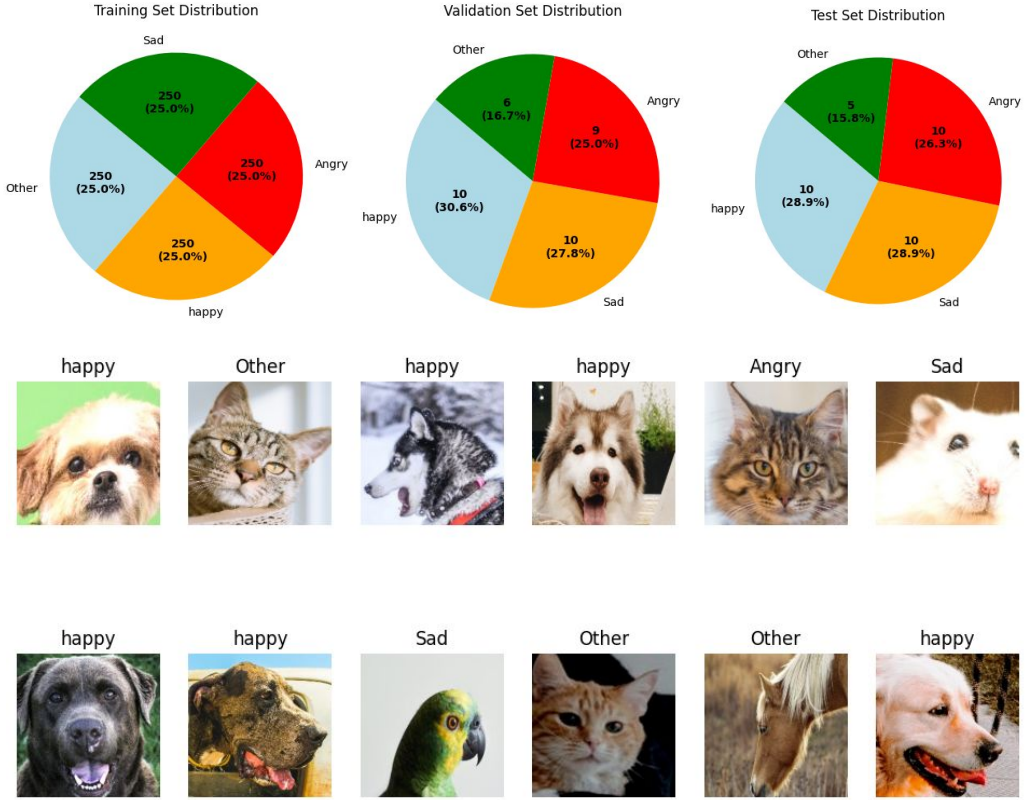
Objective:
Apply styles from reference paintings to input images

Motivation:
- Allows pet owners to transform ordinary pet photos into unique, artistic portraits
- Makes pet photos more visually appealing, memorable, and ideal for sharing
- Can be used for commercial purpose, such as designing personalized pet merchandise, greeting cards, or social media content

Model Approaches:
- Neural Style Transfer (NST)
- Fast Style Transfer (FST)
- CycleGAN

Model Evaluation:
- Neural style transfer & fast style transfer: Weighted sum of content loss and style loss
- CycleGAN: Cycle consistency loss to ensure that content and identity remain the same after the generation

EDA



Pie charts of training, validation, test set distributions

Data Size: 1000 training, 36 validation, 38 test
Class Size: 4 (Happy, Angry, Sad, Other)

Sample images with labels

Pet Types: dogs, cats, hamsters, birds, horses, rabbits, sheeps

Image Classification - Baseline Custom CNN

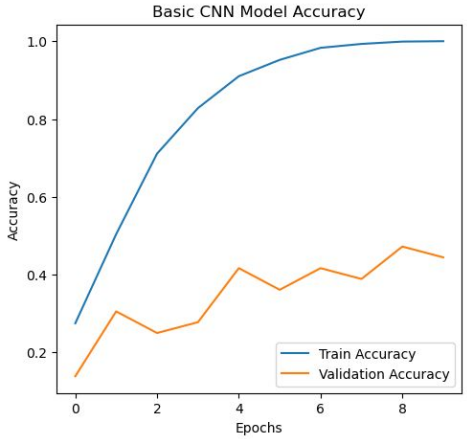
Baseline Model: Simple Custom CNN

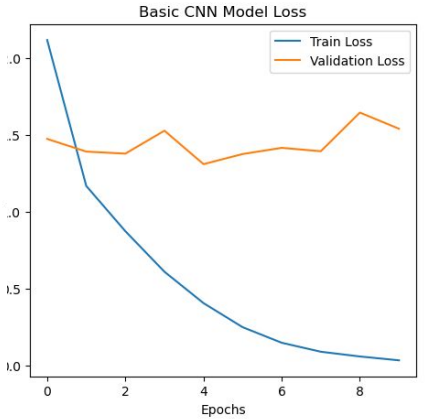
Architecture:

- Input size: (128,128,3), batch size: 32
- Single convolution layer of 32 3x3 filters with ‘ReLU’ activation
- Single max pooling layer
- Flattened layer, and two dense layers (fully connected using ReLU and output using Softmax)
- Optimizer using Adam learning rate 0.001

Total parameters: 8,129,732

	precision	recall	f1-score	support
happy	0.38	0.45	0.42	11
Sad	0.36	0.36	0.36	11
Other	0.22	0.33	0.27	6
Angry	0.20	0.10	0.13	10
accuracy			0.32	38
macro avg	0.29	0.31	0.30	38
weighted avg	0.30	0.32	0.30	38





Shortcoming of custom CNN:

- For the given small datasets, custom CNN struggles to learn meaningful patterns and generalize.
- While dropout, regularization, batchnorm can prevent overfitting, it does not compensate for lack of training data

To address overfitting and poor generalization:

1. apply data augmentation
2. use pretrained models (EfficientNet, MobileNet, ResNet)

Image Classification - Transfer Learning (ResNet)

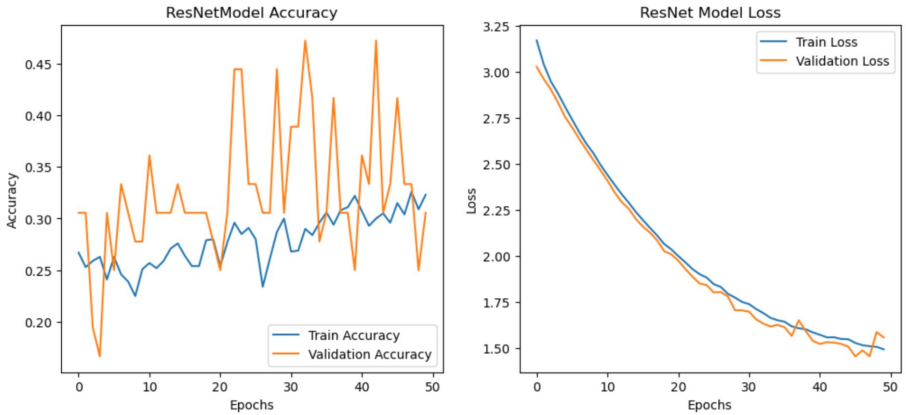
Pretrained Model: ResNet50

Incorporate Image Augmentation:

- Random rotation of +/- 40 degrees
- Random horizontal and vertical shift by up to 20%
- Random horizontal and vertical flip
- Random zoom in/out by up to 20%
- Brightness variation of between 80-120%
- Fill empty pixels with the nearest pixel value

Test set results:

	precision	recall	f1-score	support
happy	0.30	0.27	0.29	11
Sad	0.33	0.09	0.14	11
Other	0.00	0.00	0.00	6
Angry	0.32	0.80	0.46	10
accuracy			0.32	38
macro avg	0.24	0.29	0.22	38
weighted avg	0.27	0.32	0.24	38



Architecture: Input size: (128,128,3), batch size: 32. Freeze pre-trained layers with ResNet50 base model. Add two drop out layers, two dense layers, a regularization, and a learning rate scheduler.

Pros: Gap between train and validation accuracy is closing, validation loss is decreasing

Cons: Both training and validation accuracy remain quite low. Train accuracy doesn't go above 30%, Validation accuracy is erratic.

Next Steps: The relatively large size of ResNet50 might be causing overfitting due to our small dataset. Try a smaller model with shallower architectures that can help generalize better for our use case.

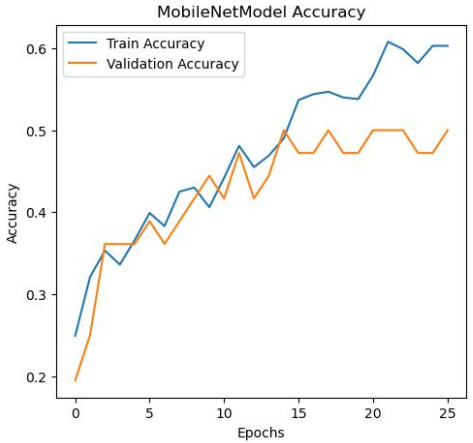
Image Classification - Transfer Learning (MobileNet)

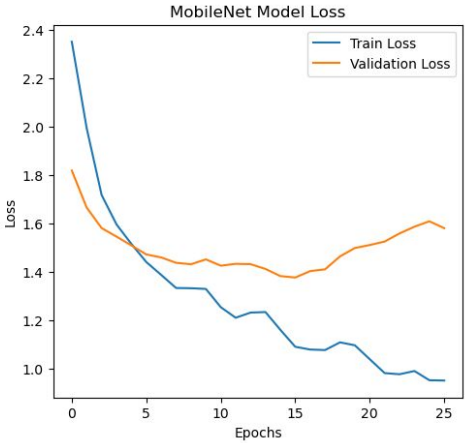
Pretrained Model: MobileNet

Preprocess/Architecture:

- Same Image augmentation
- Input size: (128,128,3), batch size: 32
- Frozen pre-trained layers with one global average pooling and two dense layers (fully connected using ReLU and output using Softmax) with l2 regularization and two dropout layers of 0.5 and 0.6 respectively
- Adam optimizer learning rate of 0.0001

Total parameters: 2,345,368





	precision	recall	f1-score	support
happy	1.00	0.36	0.53	11
Sad	0.50	0.64	0.56	11
Other	0.20	0.17	0.18	6
Angry	0.53	0.80	0.64	10
accuracy			0.53	38
macro avg	0.56	0.49	0.48	38
weighted avg	0.61	0.53	0.51	38

Compared to CNN:

- Improved generalization: better alignment between training and validation curves with improved validation accuracy between 35% to 50%
- Better convergence for both training and validation loss curves. Though the decrease for validation is subtle but overall, it is capturing meaningful patterns
- Improved test accuracy from 29% to 56% and F1 scores indicates a more balanced classification performance
- Less parameters - computationally efficient while maintaining strong performance

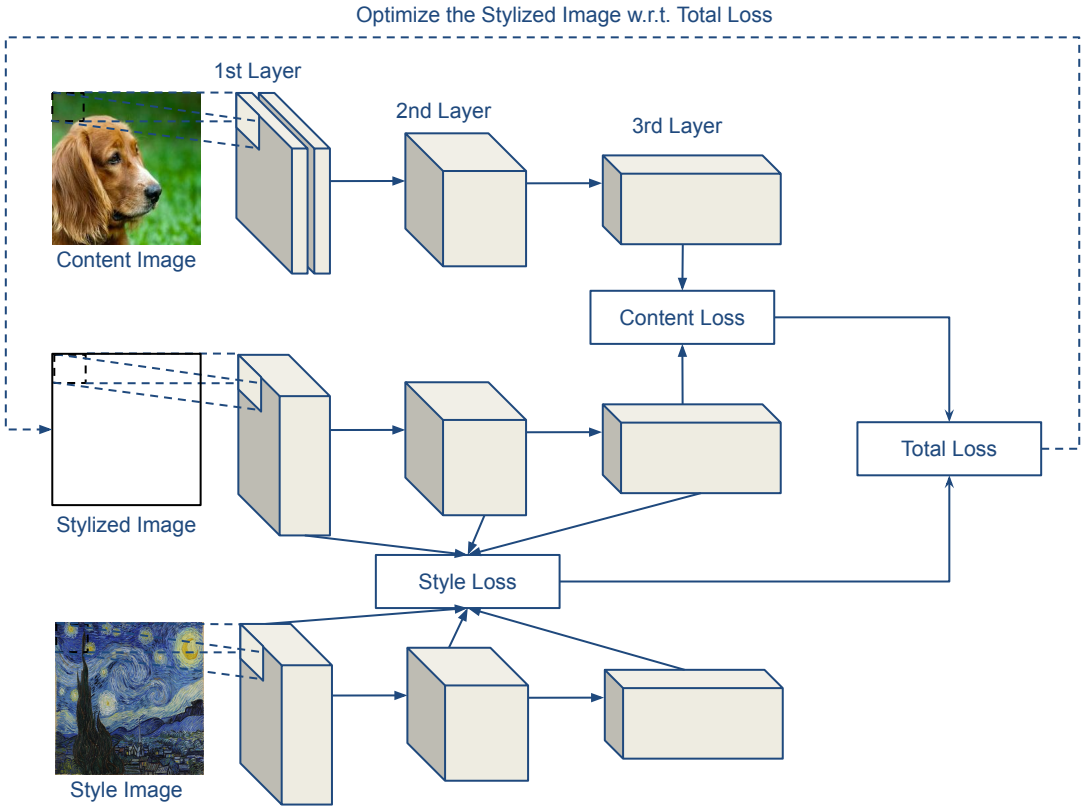
Style Transfer: Neural Style Transfer (NST)

Introduction

- Employ a pre-trained convolution neural network (CNN) to transfer styles from a given image to another
- Define a loss function that minimizes the differences between a content image, a style image and a stylized image

Loss Function

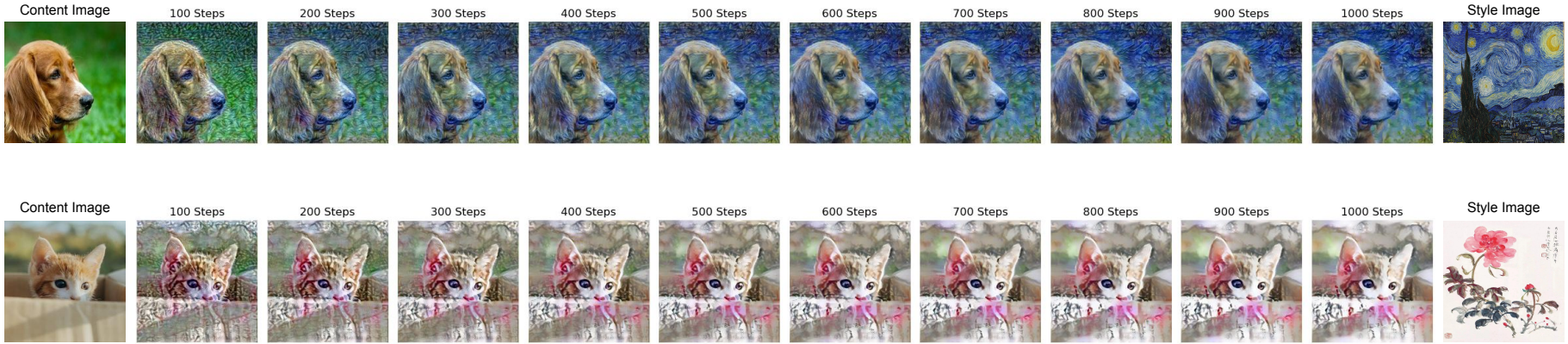
- **Content Loss:** Ensure that the activations of the higher layers are similar between the content image and the stylized image
- **Style Loss:** Ensure that the correlation of activations in all the layers are similar between the style image and the stylized image
- **Total Loss:**
  α * Content Loss + β * Style Loss



Style Transfer: Neural Style Transfer (NST)

How It Works

- Load a pre-trained CNN model (e.g. VGG19) without the classification layer
- Define the intermediate layers that represents the style and content of the image
- Build the model to extract style and content
- Run the style transfer algorithm to minimize total loss through gradient descent (e.g. Adam)
- Results:



Style Transfer: Fast Style Transfer (FST)

Introduction

- Fast artistic style transfer that may operate in real time
- Uses a pre-trained neural network to apply styles from reference paintings to input images

How It Works

- **Input:** A content image and a style image
- **Preprocessing:**
  - Decode image files into tensors
  - Resize input image and output image
  - Apply average pooling to the style image
- **Neural Network Processing:** A pre-trained model applies the style onto the content image
- **Output:** A stylized version of the content image

Pre-Trained Model

- **URL:** Google arbitrary-image-stylization-v1
- **Paper:** Exploring the structure of a real-time, arbitrary neural artistic stylization network
- Trained on a corpus of roughly 80,000 paintings
- Able to generalize to paintings previously unobserved

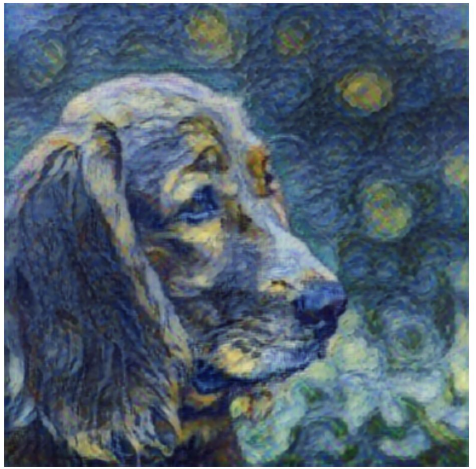
Style Transfer: Fast Style Transfer (FST)

Van Gogh Style Dog

Original Content Image

Style Image

Stylized Image



Style Transfer: Fast Style Transfer (FST)

Daqian Zhang Style Cat

Original Content Image

Style Image

Stylized Image

Style Transfer: CycleGAN

Introduction

- CycleGAN is a deep learning model for image-to-image translation without paired datasets
- It learns to transform images from one domain (Animals) to another (Paintings)
- Unlike normal supervised learning, there is no paired image (example of an exact animal matched with the artistic version)

Pre-trained Model

- **Paper:** Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
- **GitHub URL:** Cycle GAN Pix2Pix
- **Applications:** Artistic style transfer, domain adaptation

How It Works

- **Input:** An image from domain X (Animals) and an image from domain Y (Paintings)
- **Preprocessing:**
  - Normalize and resize images
  - Convert images into tensors
  - Augment images to improve robustness

Style Transfer: CycleGAN

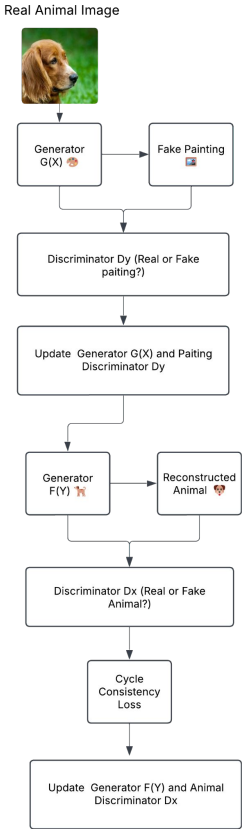
How it works (continue):

- **2 Generators:** Create an AI paintings or animals
- **2 Discriminators:** Evaluate whether an image is real or fake

Steps:

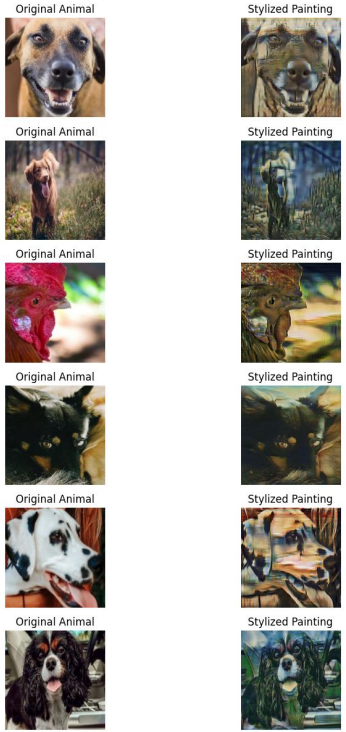
1. **Animal → Painting**
   - Generator G(X) takes real animal image and try to create a fake painting
   - Discriminator Dy evaluates if the painting is real or fake
   - G(X) is updated based on how well it fools Dy
2. **Painting → Animal**
   - The generator F(Y) takes a real painting and try to create a fake animal
   - The discriminator Dx evaluates if the animal is real or fake
   - F(Y) is updated based on how well it fools Dx
3. **Cycle Consistency Loss:** Ensure that content and identity remain the same after the generation
   - The fake painting G(X), sent back to F(Y), and it should return the original animal
   - The fake animal F(Y) is sent back to G(X) and it should return the original painting
   - If reconstruction is bad, model gets penalized
4. **Weight Updates**
   - Generator (G, F) improve to make better fake images
   - Discriminator (Dx, Dy) improve to better detect fakes

The generator keeps improving until the Discriminator can no longer tell the difference between real and fake images (Nash Equilibrium)

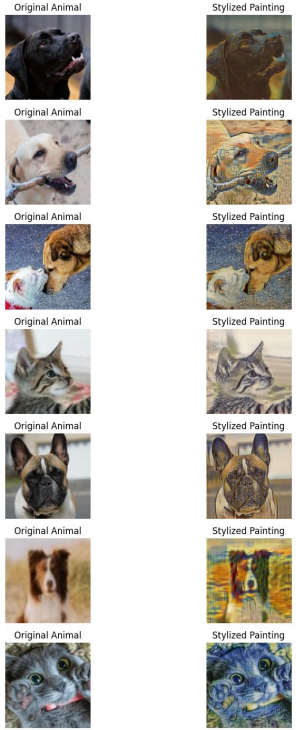


***Fake in CycleGAN means image generated by AI**

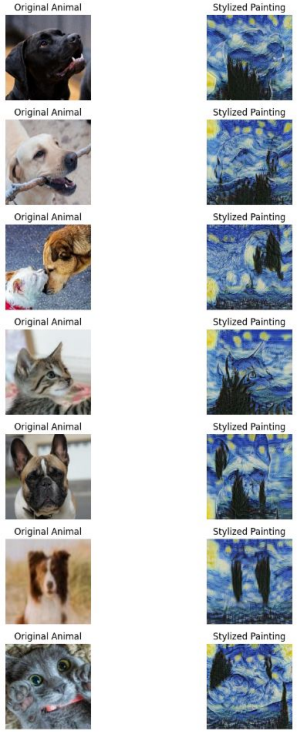
Style Transfer: CycleGAN



Train CycleGAN n_epochs = 25, n_epochs_decay = 25 with 18 paintings



Train CycleGAN n_epochs = 50, n_epochs_decay = 0, lambda_A = 15, lambda_B = 15 with 18 paintings



Train CycleGAN n_epochs = 50, with only 1 paintings

Style Transfer: Pros and Cons

Neural Style Transfer (NST)

Pros:

- **High quality result**
  Since NST optimizes for the specific content and style pair, it can produce very detailed and high-fidelity stylization
- **Fine-tunable**
  Allows customization of content/style weights for more control over the final output

Cons:

- **Computationally expensive**
  Requires multiple iterations of optimization
- **Requires careful parameter tuning**
  Adjusting settings like content and style weight is tricky. If not balanced properly, the output may look blurry or unrealistic

Fast Style Transfer (FST)

Pros:

- **Real-time processing**
  Can process images and even videos in milliseconds, making it suitable for live applications
- **Generalization**
  Uses a pre-trained model that can generalize to paintings previously unobserved

Cons:

- **Less flexible than NST**
  Cannot fine-tune the style strength dynamically as with NST
- **Lower quality for complex styles**
  While FST is fast, it might not capture intricate style details as well as NST

CycleGAN Style Transfer

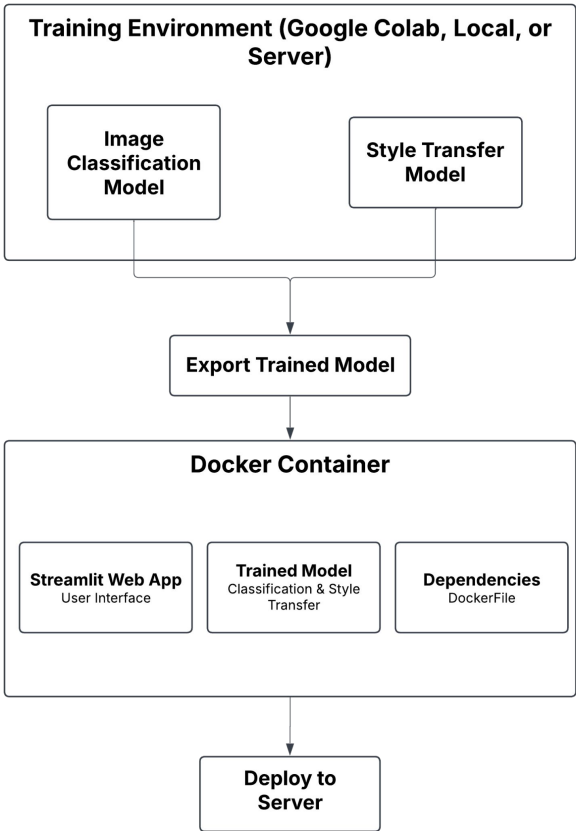
Pros:

- **Works without paired data**
- **More personalized and diverse transformations**
  It is trained to understand domain-specific style, no fixed filter
- **Realistic and high quality output**
  Because it uses the adversarial loss, to make sure the generated image looks real

Cons:

- **Very high computational cost**
  Training 1000 images with 18 paintings required 2 hours using Colab GPU
- **It is a black-box model, so it is not interpretable**

Model Operations



Model Maintenance & Parameter Update Plan:

Regular Monitoring

- Track classification accuracy & user feedback for style transfer
- Monitor processing times & error rates
- Analyze usage patterns & common failures

Data Collection Pipeline

- Implement user feedback mechanism
- Store corrected classifications & new samples

Retraining Process

- Return to training environment
- Fine-tune models with new data
- Validate improvements on test set

Deployment Updates

- Version control for models (v1, v2, v3, etc.)
- Build new Docker image with updated models
- Keep previous versions for quick rollback

Update Schedule: Monthly

Conclusion & Future Work

Image Classification (After Image Augmentation)

Models	Train Accuracy	Test Accuracy
CNN Model	0.25	0.29
EfficientNet	0.46	0.24
MobileNet	0.60	0.53
ResNet	0.32	0.32

Despite a small dataset, **MobileNet achieved the best test accuracy by 60%** for reasons:

- Depthwise separable convolution (less parameters) less prone to overfitting on small dataset
- **Pretrained weight** are more transferable, successful at **capturing image features** like edges, textures and shapes

Transfer learning significantly improved performance over a custom CNN model

Style Transfer:

- Different style transfer techniques have a **trade-off between quality and computational cost**.
- **Neural Style Transfer (NST**) provides high-quality stylized image but is computationally expensive
- **Fast Style Transfer** enables real-time processing but is less flexible
- **CycleGAN** allows **unpaired image translation** but requires high computational power.

Future Work:

- Increase training dataset by using **GANs for data augmentation** to improve model accuracy and generalization
- Expand validation dataset through **bootstrapping** or **k-fold cross-validation** for more stable result
- **Optimize classification model** through fine-tuning
- **Deploy a web app** for pet owners to predict their pet's emotional state
- **Explore hybrid models** combining Neural Style Transfer and CycleGAN for better flexibility

Thank You!

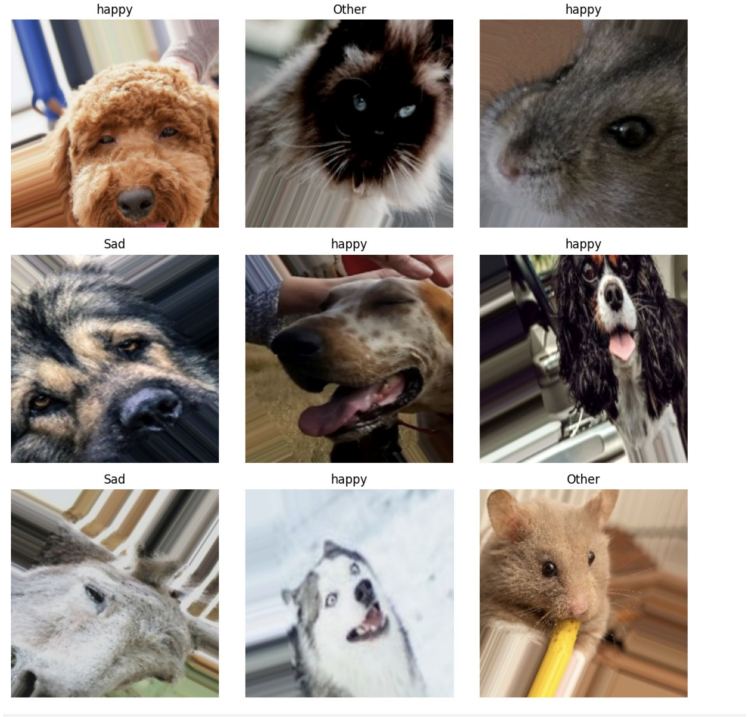
Appendix!

References

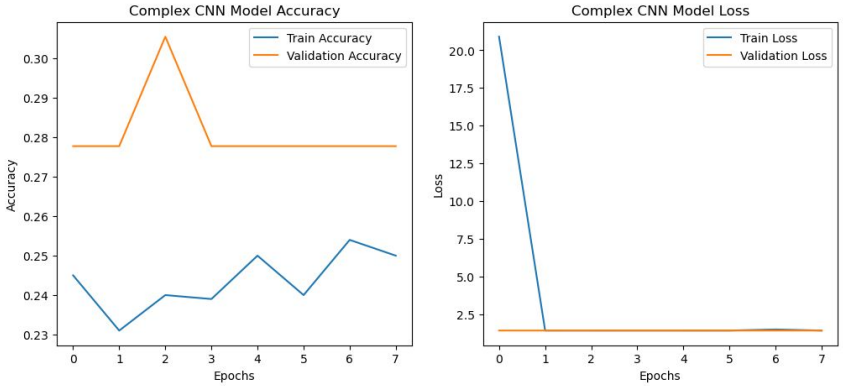
- Pet's Facial Expression Image Dataset
- Google arbitrary-image-stylization-v1 Model
- TensorFlow Neural Style Transfer
- TensorFlow Fast Style Transfer for Arbitrary Styles
- Exploring the Structure of a Real-Time, Arbitrary Neural Artistic Stylization Network
- Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
- Pytorch CycleGAN and Pix2pix

Image Augmentation

- Random rotation of +/- 40 degrees
- Random horizontal and vertical shift by up to 20%
- Random horizontal and vertical flip
- Random zoom in/out by up to 20%
- Brightness variation of between 80-120%
- Fill empty pixels with the nearest pixel value

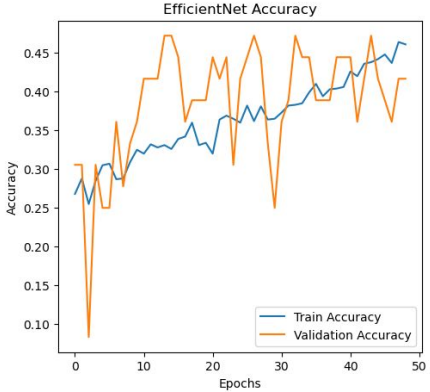


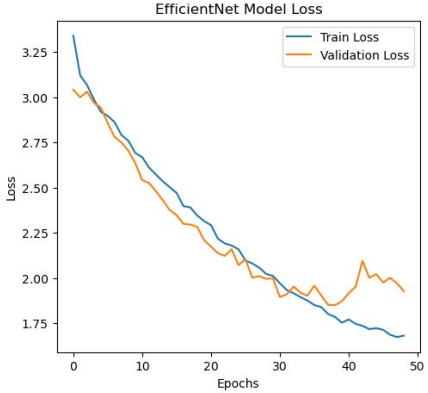
Trial and Error for Image Classification



Complex CNN Model:

We tested over 20 configurations but couldn't surpass the baseline CNN. The final model used Conv2D, Batch Normalization, MaxPooling, and Dropout. Despite early stopping, training halted at 8 epochs, with accuracy staying below 30%, showing no improvement. Building CNN model from scratch struggles to pick up meaningful signals due to the small training dataset.

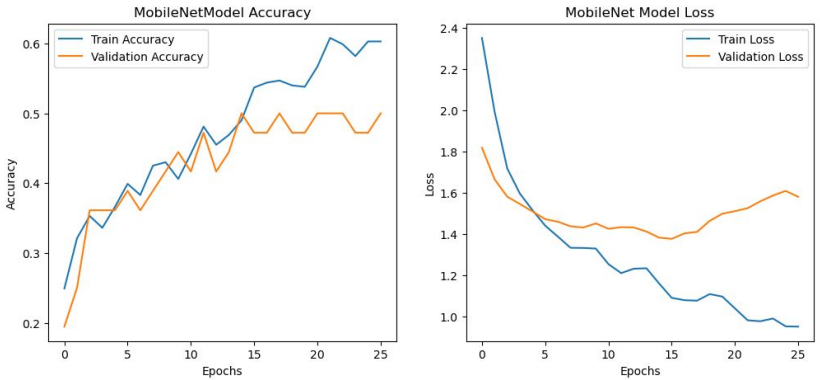


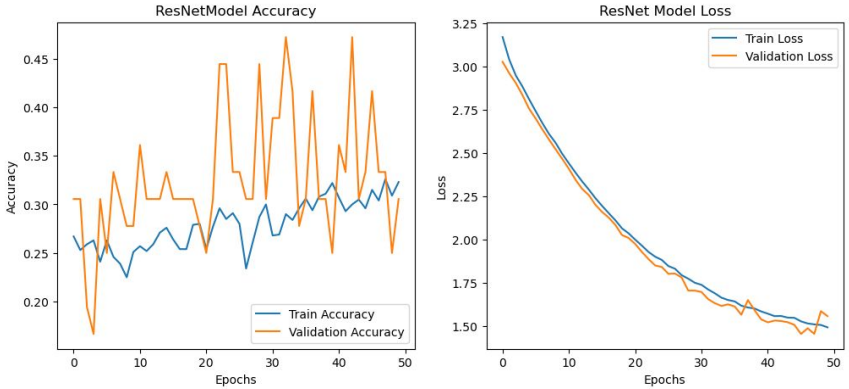


EfficientNet Transfer Learning:

While training accuracy improved from 25% to 40%, validation accuracy was inconsistent. Loss steadily decreased, but test performance was poor, likely due to overfitting on our small dataset. A smaller model may perform better.

Trial and Error for Image Classification





MobileNet Transfer Learning:

Training accuracy improved from 25% to 60%, while validation accuracy rose from 20% to 50%. Loss steadily decreased, and test performance reached 56%, the best so far. The smaller size of MobileNetV2 likely helped reduce overfitting on our small dataset.

ResNet Transfer Learning:

Train accuracy stayed around 25-30%, while validation accuracy fluctuated between 30-45%. Although the loss steadily decreased, test performance remained low at 24%, similar to the base CNN model. The large size of ResNet50 may have contributed to overfitting, especially given our small dataset.