Generating Machine Learning Models Using Machine Learning Models

A Project Report

Presented to Prof. Mark Stamp

Department of Computer Science

San Jose State University

In Partial Fulfilment

Of the Requirements for the Class

Fall-2024: CS 271

By

Aditya Patel

Karan Jain

December 2024

Abstract

This is a single paragraph, between 150-250 words in length, in which you summarize

the contents of your literature review. As you can see, the text is double-spaced. At the

end of this abstract, you will provide a list of keywords that will help readers find your

article when they are researching the same topic; these are called “Index terms” in this

paper.

***Keywords* - machine learning, convolutional neural networks, generative adversarial networks**

**Table of Contents**

[I. Introduction 1](#_heading=h.z337ya)

[A. Background 1](#_heading=h.3j2qqm3)

[B. Objective 1](#_heading=h.1y810tw)

[C. Scope 1](#_heading=h.4i7ojhp)

[II. Methodology 2](#_heading=h.2xcytpi)

[A. Hypothesis 2](#_heading=h.1ci93xb)

[B. Datasets 2](#_heading=h.3whwml4)

[C. Experimental Setup 3](#_heading=h.2bn6wsx)

[D. Models and Algorithms 4](#_heading=h.qsh70q)

[E. Implementation 7](#_heading=h.3as4poj)

[F. Evaluation Metrics 8](#_heading=h.1pxezwc)

[III. Results 10](#_heading=h.49x2ik5)

[A.](#_heading=h.2p2csry) Kernels 10

[B.](#_heading=h.147n2zr) Cosine Similarity 12

[C.](#_heading=h.23ckvvd) UMAP visualization 12

[D.](#_heading=h.ihv636) K-means Clustering and DBSCAN 13

[E.](#_heading=h.32hioqz) Observations 14

[B. Validation 17](#_heading=h.vx1227)

[IV. Discussion 17](#_heading=h.1hmsyys)

[V. Conclusion and Future Work 17](#_heading=h.41mghml)

[6. References 17](#_heading=h.2grqrue)

## I. Introduction

The potential for combining the learned features of different specialized Machine Learning (ML) models to create a third novel, task-specific model poses new challenges. Recently, diffusion models have taken the space by dazzling results. But they require huge amounts of data and processing to combine features. This project delves into solving both the issues.

Through this project, we investigate the use of a CycleGAN to merge the feature representations of two Convolutional Neural Networks (CNNs) trained on distinct domains - one on images of cats and the other on images of black objects -to create a new CNN capable of classifying black cats. The core idea hinges on the observation that the filters learned by CNNs capture domain-specific features, such as shape, texture, and color. By employing a CycleGAN - a generative adversarial network architecture designed for domain translation - we combine the filters of the two models in a way other than transfer learning, effectively transferring knowledge from different domains to generate a third model. Through our results, we see that the new CNN inherits the feature-extraction capabilities necessary for identifying black cats.

This project introduces a novel perspective on leveraging generative models like CycleGANs for ML model fusion, opening pathways for automating the creation of task-specific classifiers. By exploring the fusion of knowledge across domains, it also provides insights into the interpretability and reusability of neural network features. The methodology has implications beyond meager black cat classification, offering a framework for scarcity of labeled data as immense processing power. Through this work, we aim to contribute to the growing field of meta-learning, model-generative strategies and ML architecture design.

## II. Methodology

### A. Hypothesis

It is possible to use existing models and train a machine learning model on the learnable parameters of said models to generate a new model capable of performing a novel task that is different from the original models. In this case, merging a CNN model that detects cats and a CNN model that detects the color black will generate a CNN capable of detecting black cats.

### B. Datasets

#### 1. Dataset for CNN Model A

A collection of 1,826 images classified into two classes – ‘black’ and ‘random’. The dataset consists of 1,745 black images and 81 random images sourced from Unsplash[1] and Kaggle[2], respectively.

#### 2. Dataset for CNN Model B

A collection of 30,405 images classified into two classes – ‘cat’ and ‘random’. The dataset consists of 29,843 cat images and 562 random images sourced from Kaggle[2][3].

#### 3. Dataset for CycleGAN 1

4498 sets of 6 kernels each, obtained in the first convolutional layers after training CNN Model A and CNN Model B.

#### 4. Dataset for CycleGAN 2

4498 sets of 16 kernels each, obtained in the second convolutional layers after training CNN Model A and CNN Model B.

### C. Experimental Setup

This project conducted experiments on a Lenovo Legion Y540-15IRH laptop with the following specifications:

* **Processor:** Intel Core i5-9300H (4 cores, 8 threads, up to 4.1 GHz)
* **GPU:** NVIDIA GeForce GTX 1660 Ti with 6GB GDDR6 VRAM
* **RAM:** 16GB DDR4

The software environment was configured as follows:

* **Operating System:** Windows 11
* **Programming Language:** Python 3.11
* **Development Environment:** JupyterLab 4.3.0
* **Libraries and Frameworks:** Pytorch 2.5, torchvision, numpy, scikit-learn, seaborn, matplotlib, tqdm

JupyterLab was used as the primary development environment, providing an interactive interface for model training and evaluation. It allowed seamless execution of code cells, visualization of results, and easy experimentation with model architectures.

The Lenovo Legion Y540-15IRH laptop provided adequate computational resources, with the GPU accelerating the training and inference times.

### D. Models and Algorithms

This project utilized a combination of CNNs, CycleGANs, and clustering algorithms like One - class SVM, KMeans, and DBSCAN. The following subsections describe the architecture and hyperparameters used for each model.

#### 1. CNN Models

Convolutional neural network trained on different datasets were used to learn complimentary feature representations.

1. **Architecture**

* **Input Layer:** 32x32 pixel images with 3 channels
* **Convolutional Layers:** 2 convolutional layers with kernel size 5x5, Activation function ReLU, and max-pooling layers after each convolutional layer with pool size 2x2
* **Fully Connected Layer:** 1 dense layer with 400 neurons
* **Output Layer:** 2 neurons for binary classification

1. **Hyperparameters**

* **Optimizer:** Adam, with learning rate = 0.01
* **Batch Size:** 128
* **Epochs:** 20
* **Loss Function:** Cross-entropy loss

#### 2. CycleGAN

A variant of GAN known as CycleGAN[4] architecture was used to learn feature representations of two models using two generator models and two discriminator models. It is a framework designed to learn unpaired image-to-image translations.

1. **Architecture**

**Generator (G\_A2B and G\_B2A)**

The generator models, G\_A2B and G\_B2A, are designed to perform transformation between the domains A and B.

* **Input Channels:** 6
* **Convolutional Layers:** 2 3D convolutional layers with kernel size 3x3, Activation function ReLU after first conv3D layer and Activation function Tanh after second convo3D layer
* **Output:** Input transformed from domain A to domain B (or vice versa)

**Discriminator (D\_A and D\_B)**

The discriminator models, D\_A and D\_B are designed to distinguish real images and fake images in domains A and B respectively. Both use a PatchGAN architecture, which uses patches of images instead of whole image to determine whether sample is real or fake.

* **Input Channels:** 6
* **Convolutional Layers:** 2 3D convolutional layers with kernel size 3x3, Activation function LeakyReLU with negative slope of 0.2 after first conv3D layer and Sigmoid function after second convo3D layer to output a probability of real or fake.

1. **Hyperparameters**

* **Optimizers:** Adam, with learning rate = 0.0002, and betas = (0.5, 0.999)
* **Batch Size:** 64
* **Epochs:** 200
* **Loss Functions:** MSE Loss for Adversarial loss L1 loss for both cycle consistency loss and identity loss. Cycle consistency loss ensure generated images can be transformed back to original images.

#### 3. Generated CNN

A custom convolutional neural network for which the convolutional layer filters (kernels) are made up of filters generated by CycleGANs.

1. **Architecture**

* **Input Layer:** 32x32 pixel images with 3 channels
* **Convolutional Layers:** 2 convolutional layers with kernel size 5x5, Activation function ReLU, and max-pooling layers after each convolutional layer with pool size 2x2
* **Output:** A feature vector of length = 400

#### 4. K-means Clustering

K-means clustering was used to analyze the feature vectors, and to group the vectors into clusters.

1. **Hyperparameters:**

* **Number of clusters ( k ):** 2 or 3

#### 5. DBSCAN

DBSCAN was used to group feature vectors generated by the CNN for test images into clusters.

1. **Hyperparameters:**

* **Epsilon:** 3
* **Min Samples:** 3

**6. UMAP**

UMAP was used to construct a weighted graph representing the high-dimensional feature vectors and (thereby reduce dimensionality), where each point was connected to its nearest neighbors based on a chosen distance metric.(different for each model tested).

**7. One Class SVM :**

One-Class SVM was used to learn a decision boundary around the feature vectors and identify whether new points belong to this class or are outliers.

**a.Hyperparameters:**

* **kernel='rbf'**
* **nu=0.1**
* **gamma='scale'**

### E. Implementation

The implementation steps involved were as follows:

1. Train 1749 CNNs on the ‘cat’ dataset and 2749 CNNs on the ‘black’ dataset.
2. Train two CycleGANs to learn and merge feature representations from the original models, one for each convolutional layer.
3. Create the third model by using the generated kernels from each CycleGAN.
4. Perform clustering on the feature vectors obtained by passing test images to the generated model.
5. Evaluate and study the clusters.

### F. Evaluation Metrics

#### 1. Cluster Entropy

Cluster entropy is a measure of randomness or uncertainty in the cluster. Lower entropy values mean that cluster contains datapoints from the same class.

H(Ck) = - M∑j=1 pj,k log2(pj,k)

Where M is number of classes, pj,k is the probability that points in cluster Ck belong to class j. The overall cluster entropy is the weighted average of individual entropies of all clusters.

H = K∑k=1 nk H(Ck)

#### 2. Cluster Purity

Cluster purity is used to measure the quality of clusters produced. It is a simple measure of cluster homogeneity. High purity indicates more data points in a cluster belong to the same class.

U(Ck) = maxj nj,k  
 nj,k

#### 3. Accuracy

Accuracy measures the number of correct predictions out of total predictions. It can be calculated as follows:

Accuracy = True Positive (TP) + True Negative (TN)Total Samples Accuracy =

#### 4. Precision

Precision is the ratio of True positives and total number of points classified as positive. It can be calculated as follows:

Precision =

#### 5. Recall

Recall is the ratio of corrected predicted positive instances out of all actual positive instances. It can be calculated as follows:

Recall =

## III. Results

### Kernels

A group of squares with numbers

Description automatically generated

Kernels of CNN Model A and the kernels generated from them using CycleGAN

A screenshot of a graph

Description automatically generated

Kernels of CNN Model B and the kernels generated from them using CycleGAN

### Cosine Similarity

### A blue and red grid with red lines Description automatically generated

Cosine similarity heat map of feature vectors obtained from Generated Model 229

### UMAP visualization

A graph with numbers and dots

Description automatically generated

UMAP visualization of feature vectors obtained from Generated Model 229

### K-means Clustering and DBSCAN

A diagram of a graph

Description automatically generated with medium confidence

A chart with a number of dots

Description automatically generated with medium confidence

### Observations

|  |  | Kmeans | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| Model | Generated Filter Domain | Accuracy | Precision | Recall | Entropy | Purity |
| 229 | A | 0.876 | 0.852 | 0.828 | 0.537 | 0.876 |
| 1486 | A | 0.876 | 0.823 | 0.848 | 0.528 | 0.876 |
| 1234 | A | 0.842 | 0.911 | 0.738 | 0.61 | 0.842 |
| 1330 | A | 0.842 | 0.735 | 0.833 | 0.59 | 0.842 |
| 1304 | A | 0.82 | 0.852 | 0.725 | 0.672 | 0.82 |
| 1314 | A | 0.82 | 0.9411 | 0.695 | 0.629 | 0.82 |
| 226 | A | 0.808 | 0.588 | 0.869 | 0.562 | 0.809 |
| 307 | A | 0.797 | 0.558 | 0.863 | 0.566 | 0.797 |
| 229 | B | 0.797 | 0.823 | 0.7 | 0.724 | 0.797 |
| 478 | B | 0.797 | 0.588 | 0.833 | 0.605 | 0.797 |
| 805 | B | 0.797 | 0.764 | 0.722 | 0.723 | 0.797 |
| 1291 | B | 0.797 | 0.676 | 0.766 | 0.686 | 0.797 |
| 1205 | A | 0.786 | 0.617 | 0.777 | 0.673 | 0.786 |
| 1567 | A | 0.786 | 0.617 | 0.777 | 0.673 | 0.786 |
| 311 | B | 0.786 | 0.647 | 0.758 | 0.697 | 0.786 |
| 253 | A | 0.775 | 0.705 | 0.705 | 0.756 | 0.775 |
| 1215 | B | 0.764 | 0.882 | 0.638 | 0.75 | 0.764 |
| 1278 | A | 0.752 | 0.911 | 0.62 | 0.739 | 0.752 |
| 1703 | A | 0.752 | 0.705 | 0.666 | 0.801 | 0.752 |
| 214 | A | 0.741 | 0.676 | 0.657 | 0.814 | 0.741 |
| 959 | A | 0.741 | 0.529 | 0.72 | 0.72 | 0.741 |
| 282 | B | 0.741 | 0.794 | 0.627 | 0.817 | 0.741 |
| 1470 | A | 0.73 | 0.735 | 0.625 | 0.84 | 0.73 |
| 1507 | A | 0.73 | 0.794 | 0.613 | 0.831 | 0.73 |
| 1311 | A | 0.696 | 0.823 | 0.571 | 0.849 | 0.696 |
| 1337 | A | 0.696 | 0.911 | 0.563 | 0.775 | 0.696 |
| 1540 | B | 0.606 | 0.735 | 0.49 | 0.935 | 0.606 |

Clustering feature vectors using k-means.

|  |  | One -class SVM | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| Model | Generated Filter Domain | Accuracy | Precision | Recall | Entropy | Purity |
|  |  |  |  |  |  |  |
| 1304 | A | 0.82 | 0.882 | 0.714 | 0.667 | 0.82 |
| 1314 | A | 0.764 | 0.852 | 0.644 | 0.767 | 0.764 |
| 229 | A | 0.752 | 0.911 | 0.62 | 0.739 | 0.752 |
| 1337 | A | 0.741 | 0.941 | 0.603 | 0.716 | 0.741 |
| 1486 | A | 0.741 | 0.941 | 0.603 | 0.716 | 0.741 |
| 1234 | A | 0.73 | 0.911 | 0.596 | 0.757 | 0.73 |
| 1278 | A | 0.73 | 0.911 | 0.596 | 0.757 | 0.73 |
| 1507 | A | 0.719 | 0.9411 | 0.581 | 0.729 | 0.719 |
| 1205 | A | 0.707 | 0.911 | 0.574 | 0.7704 | 0.707 |
| 282 | B | 0.685 | 0.911 | 0.553 | 0.778 | 0.685 |
| 1291 | B | 0.685 | 0.911 | 0.553 | 0.778 | 0.685 |
| 1311 | A | 0.64 | 0.911 | 0.516 | 0.781 | 0.674 |
| 253 | A | 0.629 | 0.911 | 0.508 | 0.778 | 0.685 |
| 805 | B | 0.5955 | 0.9411 | 0.484 | 0.716 | 0.741 |
| 1215 | B | 0.55 | 0.911 | 0.455 | 0.728 | 0.764 |
| 1703 | A | 0.505 | 0.941 | 0.432 | 0.61 | 0.831 |
| 307 | A | 0.494 | 0.9411 | 0.426 | 0.591 | 0.842 |
| 214 | A | 0.449 | 0.911 | 0.402 | 0.561 | 0.865 |
| 959 | A | 0.426 | 0.882 | 0.389 | 0.569 | 0.865 |
| 1540 | B | 0.426 | 0.882 | 0.389 | 0.569 | 0.865 |
| 311 | B | 0.415 | 0.911 | 0.387 | 0.471 | 0.898 |
| 229 | B | 0.404 | 0.941 | 0.385 | 0.355 | 0.932 |
| 1470 | A | 0.382 | 0.941 | 0.376 | 0.262 | 0.955 |
| 226 | A | 0.359 | 0.882 | 0.361 | 0.338 | 0.932 |
| 1330 | A | 0.359 | 0.911 | 0.364 | 0.245 | 0.955 |
| 1567 | A | 0.359 | 0.911 | 0.364 | 0.245 | 0.955 |
| 478 | B | 0.359 | 0.911 | 0.364 | 0.245 | 0.955 |

Classifying feature vectors using One-class SVM.

|  |  | DBSCAN | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| Model | Generated Filter Domain | Accuracy | Precision | Recall | Entropy | Purity |
|  |  |  |  |  |  |  |
| 229 | A | 0.887 | 0.852 | 0.852 | 0.501 | 0.887 |
| 1486 | A | 0.887 | 0.823 | 0.875 | 0.568 | 0.876 |
| 282 | B | 0.865 | 0.764 | 0.866 | 1.926 | 0.573 |
| 229 | B | 0.853 | 0.705 | 0.888 | 1.281 | 0.707 |
| 1234 | A | 0.842 | 0.911 | 0.738 | 0.61 | 0.842 |
| 1330 | A | 0.842 | 0.705 | 0.857 | 0.698 | 0.831 |
| 1470 | A | 0.842 | 0.647 | 0.916 | 1.5 | 0.606 |
| 1540 | B | 0.842 | 0.647 | 0.916 | 1.793 | 0.483 |
| 1314 | A | 0.831 | 0.882 | 0.731 | 1.396 | 0.662 |
| 1304 | A | 0.82 | 0.852 | 0.725 | 0.794 | 0.808 |
| 226 | A | 0.808 | 0.588 | 0.869 | 0.562 | 0.809 |
| 478 | B | 0.797 | 0.588 | 0.833 | 1.178 | 0.617 |
| 805 | B | 0.797 | 0.764 | 0.722 | 0.723 | 0.797 |
| 307 | A | 0.786 | 0.529 | 0.857 | 0.569 | 0.786 |
| 1567 | A | 0.786 | 0.617 | 0.777 | 0.673 | 0.786 |
| 1703 | A | 0.786 | 0.676 | 0.741 | 1.268 | 0.707 |
| 311 | B | 0.786 | 0.647 | 0.758 | 1.292 | 0.528 |
| 1291 | B | 0.786 | 0.588 | 0.8 | 1.62 | 0.573 |
| 253 | A | 0.775 | 0.705 | 0.705 | 0.756 | 0.775 |
| 1507 | A | 0.775 | 0.529 | 0.818 | 1.737 | 0.5584 |
| 214 | A | 0.764 | 0.676 | 0.696 | 0.769 | 0.764 |
| 1205 | A | 0.752 | 0.5 | 0.772 | 0.653 | 0.752 |
| 959 | A | 0.741 | 0.529 | 0.72 | 0.781 | 0.741 |
| 1311 | A | 0.719 | 0.617 | 0.636 | 1.777 | 0.539 |
| 1337 | A | 0.696 | 0.911 | 0.563 | 0.877 | 0.685 |
| 1278 | A | 0.674 | 0.676 | 0.56 | 1.818 | 0.46 |
| 1215 | B | 0.606 | 0.294 | 0.476 | 2.017 | 0.449 |

Clustering feature vectors using DBSCAN

## IV. Discussion

With this project, we used CNN’s as feature extractors to recognize and differentiate images. After extracting features, we applied dimensionality reduction(UMAP) and clustered similar results together with K-means and DBScan . We find that the black cat images were clustered together effectively demonstrating unsupervised learning, i.e. the generated CNN was able to differentiate black cat images from others without being trained on any data.

Metrics such as cosine similarity further validated the ability of CNN to produce meaningful features. This highlights the potential of such a technique to perform well even with scarce data and computation power.

A few challenges observed during the project were that of hyperparameter selection and dimensionality of the feature vectors, which impacted the clustering results significantly. Additionally, the translation of filters from one domain to another was abrupt due to diverse dataset, thereby further affecting the performance of the CNN as a feature extractor.

## V. Conclusion and Future Work

The project demonstrates that new models can be generated by combining the features of other similar machine learning models. It is possible to use existing models and train a machine learning model on the learnable parameters of said models to generate a new model capable of performing a novel task that is different from the original models. In this case, merging a CNN model that detects cats and a CNN model that detects the color black will generate a CNN capable of detecting black cats.

By combining the extracted features with clustering algorithms and dimensionality reduction techniques like UMAP we showed that CNN can learn in an unsupervised manner.

Several directions for future work can be highlighted

* Hyper-parameter Optimization for Clustering
* Feature space exploration with alternative metrics and feature weighting
* Semi Supervised Learning with small dataset
* Improving domain translation
* End to end pipeline development based on semantics

## 6. References

| [1] | https://unsplash.com/s/photos/black |
| --- | --- |
| [2] | https://www.kaggle.com/datasets/ezzzio/random-images |
| [3] | https://www.kaggle.com/datasets/borhanitrash/cat-dataset/data |
| [4] | 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. 2242-2251, doi: 10.1109/ICCV.2017.244. |