TRANSFORMING Image Recognition and Classification

(Using CIFAR 100 & Google Landmarks Dataset v2)

Spring 2023 Term: W281 Computer Vision Final Project Report

Waqas Ali | Pedro Melendez | Prakash Krishnan

1. Abstract

Image classification is an essential task in computer vision. There has been considerable research using traditional handcrafted feature extraction techniques with classical machine learning models and now deep learning techniques. Image classification has several critical use cases, such as in self-driving cars, medical image classification, and satellite imaging. This paper explores using a Pre-trained Vision Transformer that is fine-tuned on a downstream task for image recognition and classification. Self-attention-based Transformer architectures have become the model of choice in natural language processing. In this study, we modify a traditional transformer architecture for image classification. We compare the experimental results with the handcrafted techniques and CNN models. The paper concludes by providing insights into future studies and potential use cases of Vision-Language Models that combine Computer Vision and Natural Language Processing.

2. Introduction

The last two decades have seen an explosion in the amount of video and multimedia digital data being generated, stored, transmitted, analyzed, and accessed. Advances in digital image processing techniques, data repository technologies, smartphones, and cameras have driven this. Retrieving and classifying from this vast body of data is challenging and complex.

Our inspiration for this project came about from two research papers (1) An Image is worth 16x16 words - Transformers for Image Recognition at Scale (Alexey Dosovitskiy et al., ICLR 2021) (2) Google Landmarks Dataset v2 (Tobias Weyand et al., 2020)

In the area of deep learning, the use of transfer learning to pre-train a neural network on a large dataset and then fine-tune the model on the specific dataset is now the common approach. CNN models such as Resnet, VGGnet, GoogLenet, and Alexnet have been successfully deployed in CV applications. Prior to the deployment of Vision Transformers in Computer Vision, CNN models were deemed state-of-the-art.

Vision Transformers (ViT) are challenging that paradigm. Self-attention-based Transformer architectures have become the model of choice in natural language processing. The approach is to pre-train on a large corpus of data and fine-tune it on a smaller dataset. Transformers are highly scalable and computationally efficient. The research paper cited in this report describes a technique to adapt a traditional transformer architecture for images.

GLDv2 is an exciting and challenging dataset that includes human-made and natural landmarks contributed to Wikimedia Commons. It was designed to simulate real-world conditions and, thus, many practical problems. However, GLDv2 is resource and computation-heavy.

For this project, we started our study on the CIFAR 100 dataset (Canadian Institute for Advanced Research, 100 classes), consisting of 60,000 32x32 color images with 50,000 training and 10,000 test images. Additionally, to assess scalability and generalizability, we complemented our study by building our models with a subset of the GLDv2

dataset but included only 20 classes with a total of 16,633 images that had a much better resolution; most images were 800x800.

Feature extraction has been a critical component of a CV pipeline, and handcrafted techniques such as Histogram of Gradients (HOG), Gray Level Co-occurrence Matrices (GLCMs), Scale-invariant feature transform (SIFT), Oriented Fast and Rotated Brief (ORB) and Speeded-Up Robust Feature (SURF) are popular. The extracted features are then fed into an ML classifier, such as Support Vector Machine (SVM) or Logistic Regression, to predict a classification output.

Deep learning (DL) neural networks, particularly pre-trained models, do not manually extract features from an image but extract the features through learning to offer state-of-the-art results over traditional handcrafted techniques.

Our hypothesis is that Vision Transformers sufficiently pre-trained on a large corpus of images and fine-tuned should perform just as well or even better than the CNN models. We perform a variety of experiments to validate our hypothesis.

3. Executive Summary

We had three research questions for our image recognition and classification study.

1. How well does a Pre-Trained Vision Transformer Model (ViT) that is Fine Tuned compare to CNN models (ResNet 9 & ResNet 50) and models built on manual feature extraction techniques (HOG+SVM, SIFT+SVM, GLCM+SVM)?
2. How does the state-of-the-art ViT model generalize and scale?
3. What is the future of ViTs?

The summary results are presented below in Figure 3.1.

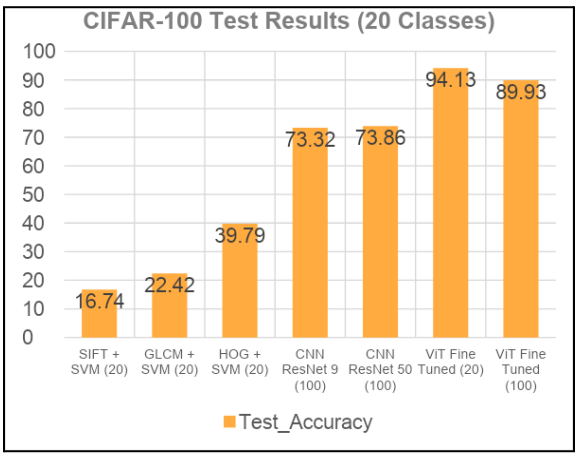


Figure 3.1: Summary Results on CIFAR-100

Vision Transformer outperformed CNN and traditional CV methods and achieved a test accuracy of 94.13% in 20 classes. *What is more remarkable is that with 100 classes on the CIFAR-100 dataset, it achieved an 89.93% accuracy.*

The misclassifications were primarily due to class categories not being distinctive as in Vehices_1 and Vehicles_2 and image resolution of only 32 by 32 for feature extraction. For the 20-class model, the class accuracy ranged from 89.6% on the low end to 98.4% on the high end. See the confusion matrix below for the ViT Model (Table 3.1).

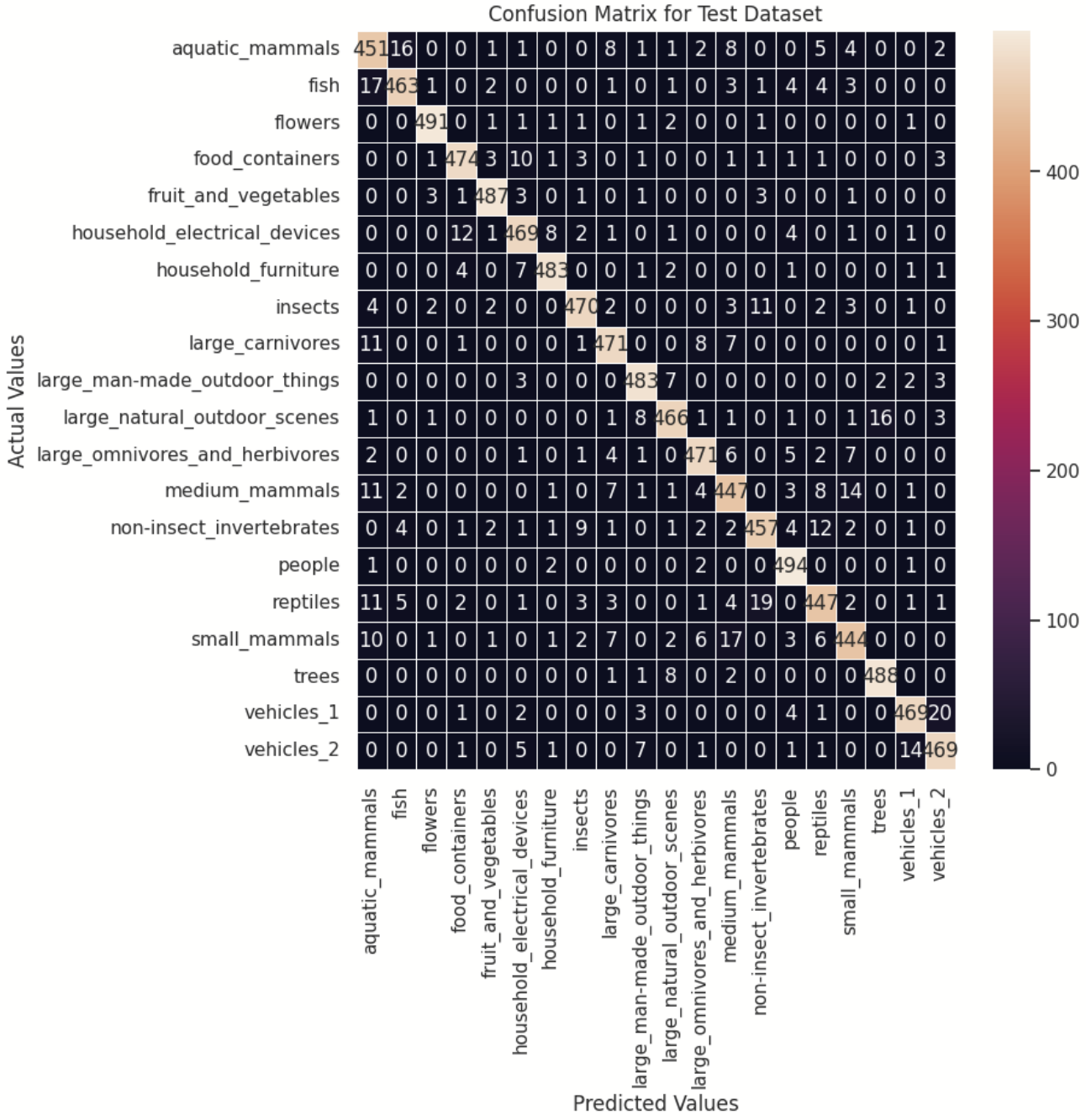


Table 3.1: Confusion Matrix for the ViT Model on CIFAR-100

The ViT model also generalized well on the GLDv2 dataset with a test accuracy of 98% (See Figure 3.2). Furthermore, the ViT model correctly classified low-detailed sketches and full-colored drawings. (See Section 11).

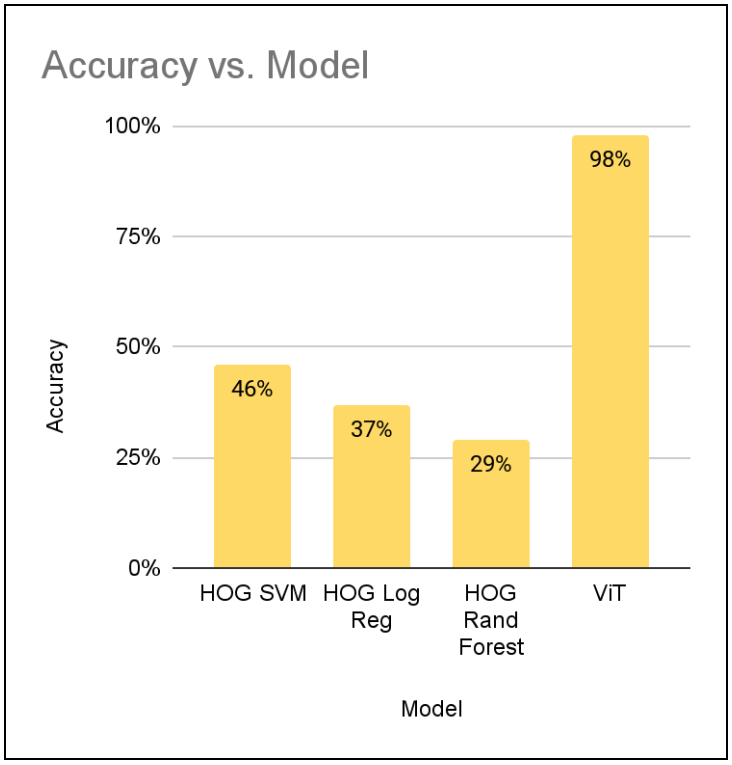


Table 3.2: Test Results on GLDv2

This study supports our assertion that Vision Transformers have a significant role in CV Classification and have an opportunity to challenge CNN as the preferred neural network architecture.

All supporting documents and Jupyter notebooks are included in our team's GitHub repository: https://github.com/elpeme/w281_final_project

4. About Dataset

CIFAR-100

CIFAR 100 Dataset (https://www.cs.toronto.edu/~kriz/cifar.html) is used for model building and evaluation. CIFAR-100 is a popular dataset for image classification studies and is widely used for model evaluation and testing. This dataset has 100 classes (fine labels) containing 600 images each. There are 500 training images and 100 testing images per class. The 100 classes in the CIFAR-100 are grouped into 20 superclasses (course labels). Each image comes with a "fine" label (the class to which it belongs) and a "coarse" label (the superclass to which it belongs).

See Table 4.1 for a mapping of CIFAR-100 coarse to fine label mapping.

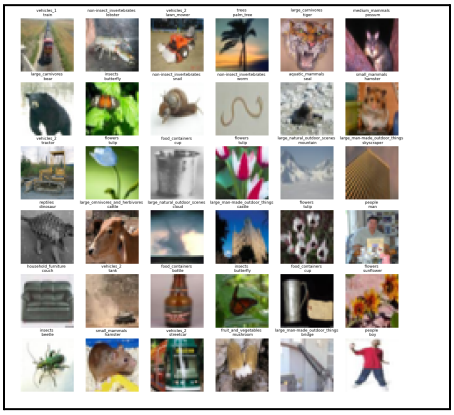
Table 4.1: CIFAR-100 Coarse and Fine Label Mapping

Coarse_Label_Name	Fine_Label_Name
aquatic_mammals	[otter, seal, whale, beaver, dolphin]
fish	[aquarium_fish, shark, flatfish, ray, trout]
flowers	[sunflower, rose, tulip, poppy, orchid]
food_containers	[cup, bottle, plate, bowl, can]
fruit_and_vegetables	[apple, mushroom, sweet_pepper, orange, pear]
household_electrical_devices	[telephone, keyboard, television, clock, lamp]
household_furniture	[table, chair, wardrobe, couch, bed]
insects	[cockroach, butterfly, bee, caterpillar, beetle]
large_carnivores	[wolf, leopard, lion, tiger, bear]
large_man-made_outdoor_things	[castle, skyscraper, road, bridge, house]
large_natural_outdoor_scenes	[cloud, sea, mountain, forest, plain]
large_omnivores_and_herbivores	[cattle, elephant, chimpanzee, camel, kangaroo]
medium_mammals	[possum, skunk, raccoon, fox, porcupine]
non-insect_invertebrates	[lobster, snail, worm, cra, spider]
people	[boy, woman, girl, man, baby]
reptiles	[dinosaur, snake, crocodile, turtle, lizard]
small_mammals	[squirrel, shrew, rabbit, hamster, mouse]
trees	[willow_tree, pine_tree, oak_tree, maple_tree,...
vehicles_1	[train, bicycle, motorcycle, bus, pickup_truck]
vehicles_2	[streetcar, tractor, rocket, tank, lawn_mower]

The data is curated, clean, and complete. There were no missing values, and all the images had the same size in RGB. The only pre-processing that was needed was to convert to grayscale and floating point format for the manual feature extraction techniques leveraging OpenCV.

A sampling of images is shown below in Figure 4.1

Figure 4.1: Sample Images from CIFAR-100 with Coarse and Fine Labels



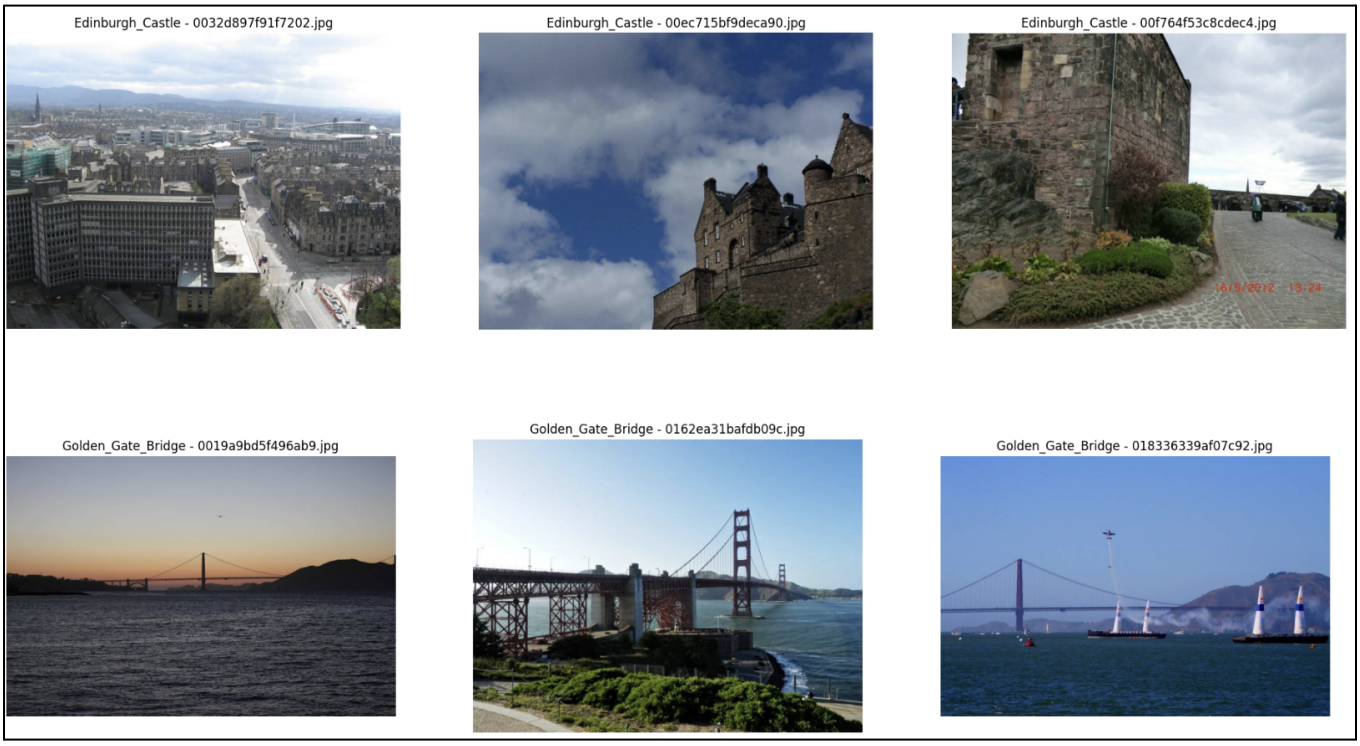
GLDv2 Dataset

The full GLDv2 dataset (https://github.com/cvdfoundation/google-landmark) is the largest dataset in the realm of human-made and natural landmarks, it contains over 5 million images and 200,000 distinct instance labels.

For our study, we used a subset of GLDv2 provided by Kaggle for the 2021 Landmark competition (Google Landmark Recognition 2021 | Kaggle), which consisted of 1,580,470 images and 81,313 classes.

A sampling of images from the dataset is shown below in Figure 4.2.

Figure 4.2: Sample Images from GLDv2



5. Exploratory Data Analysis

The CIFAR-100 dataset has 50,000 examples in the training dataset and 10,000 examples in the test dataset.

Each example has the following fields ['img', 'fine_label', 'coarse_label'].

- Image (img) is a PIL object, RGB and 32x32
- Fine Label (fine_label) is an integer code for the fine label class ranging from 0-99
- Coarse Label (coarse_label) is an integer code for the coarse label class ranging from 0-19

The distribution of the training dataset across the coarse label class is as follows:

Table 5.1: CIFAR-100 Training Example Distribution by Coarse Label Class

	Image_Count
aquatic_mammals	2500
fish	2500
flowers	2500
food_containers	2500
fruit_and_vegetables	2500
household_electrical_devices	2500
household_furniture	2500
insects	2500
large_carnivores	2500
large_man-made_outdoor_things	2500
large_natural_outdoor_scenes	2500
large_omnivores_and_herbivores	2500
medium_mammals	2500
non-insect_invertebrates	2500
people	2500
reptiles	2500
small_mammals	2500
trees	2500
vehicles_1	2500
vehicles_2	2500

Table 5.1 shows that there is no class imbalance, as the distribution is equal across the course classes. Figure 5.1 provides an inspection sample of images from the training dataset.

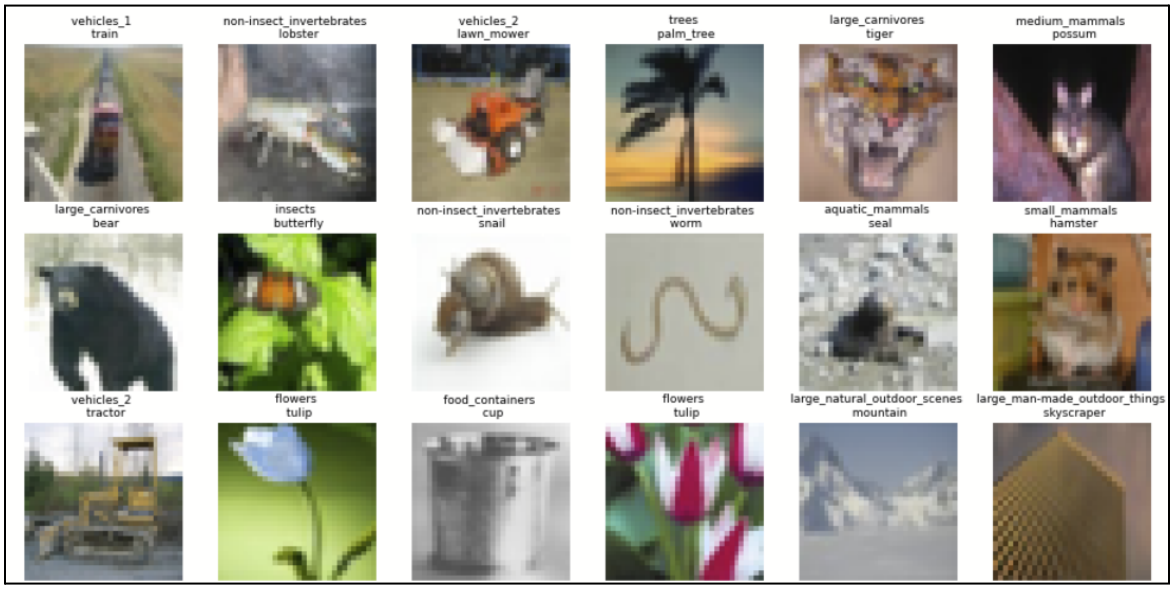


Figure 5.1: Samples Train Images from CIFAR-100 Dataset

The GLDv2 Kaggle dataset is heavily unbalanced, with most classes containing less than 5 images and 4 outliers classes with more than 1000 images each. We decided to discard the outlier classes with over 1000 images and took the top 20 categories to ensure that all classes were relatively balanced. Therefore, we ended up with a dataset with 20 classes and 16,633 images. Additionally, to overcome computational challenges, we blurred and downsized the images to reach a resolution of 224x224.

Table 5.2: GLDv2 Training Example Distribution by Category

	category
Khotyn_Fortress	971
Pe%C5%A1ter	966
Niagara_Falls	944
Haleakal%C4%81_National_Park	920
Feroz_Shah_Kotla	900
Golden_Gate_Bridge	888
Grand_Canyon	888
Catedral_San_Sebasti%C3%A1n,_Cochabamba	888
Madrid_R%C3%ADo	861
Kecharis	839
Hayravank_monastery	820
Mathura_Museum	820
Qutb_Minar_and_its_monuments,_Delhi	776
Sofiyivsky_Park	757
St._Lawrence,_Toronto	754
Akkerman_fortress	742
Edinburgh_Castle	734
Eiffel_Tower	731
Genoese_fortress_(Sudak)	730
Skopje_Fortress	704

6. Feature Extraction Techniques

In this section, we describe the use of several manual feature extraction techniques. We randomly selected an image from five different course categories to assess the efficacy of manual feature extraction techniques. These images represented a variety of objects with distinct edges, contours, blobs, textures, and corners. The five select images' color and grayscale images are shown below in Figure 6.1.

Figure 6.1: Gray and Color Images for Each of the Five Coarse Categories

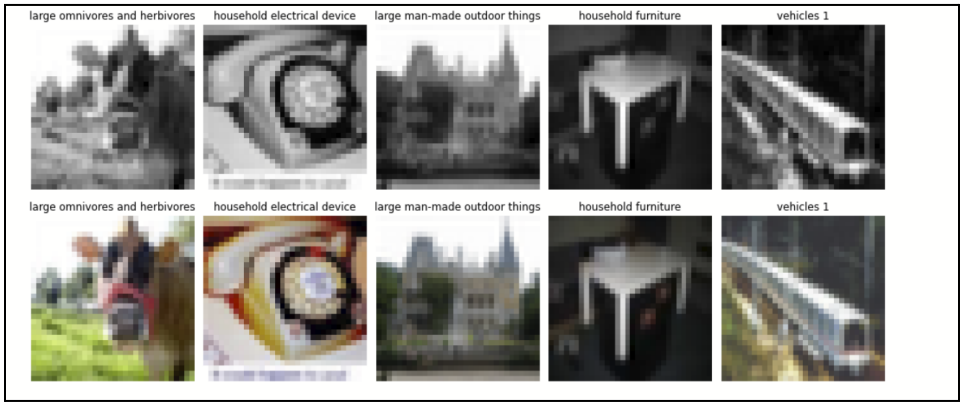


Figure 6.1 shows that we need a robust feature detection algorithm that can detect corners, textures, edges, blobs, and ridges. A single feature extraction technique may not appropriately detect the features when dealing with 100 fine classes and 20 coarse classes as in the CIFAR-100 dataset.

The traditional CV techniques relying on extracting the image features manually can be summarized as follows (see Table 6.1):

Table 6.1: Overview of Manual Feature Extraction Techniques

Item	Technique	Typical Application
1.	Histogram of Oriented Gradients (HOG)	Detecting Edges
2.	Harris Corner	Detecting Corners
3.	Scale-Invariant Feature Transform (SIFT)	Detecting Blobs
4.	SURF	Detecting Blobs
5.	Features from Accelerated Segment Test (FAST)	Detecting Corners
6.	Binary Robust Independent Elementary Features (BRIEF)	Detecting Blobs
7.	Oriented FAST and Rotated BRIEF (ORB)	Combination of Corners and Blobs
8.	Gray Level Co-occurrence Matrices (GLCM)	Detecting Textures

For this project, we will explore the use of **HOG, SIFT, and GLCM** to detect key points and generate a feature vector that can be fed into a classifier for training and classification. Traditional manual CV techniques are only good at certain feature aspects, as shown above. The manual techniques do not scale well and require domain knowledge.

As we have images with a combination of features, we need a neural network that is trained on a large corpus of images and can handle a variety of image features such as corners, edges, blobs, contours, and ridges automatically. These neural networks perform feature extraction and classification that is end-to-end trainable and deemed state-of-the-art.

Convolutional Neural Networks (CNN) such as AlexNet, and VGGNet have been popular with simple to medium complexity problems while deeper networks like Inception and ResNet have been deployed for more complex applications. CNNs perform automatic feature extraction and classification.

For this project, we will explore the use of **ResNet 9 and ResNet 50** for feature extraction and classification.

Finally, given the computational efficiency and scalability of Transformers, we explore the use of a Vision Transformer (**ViT**) and compare the results with traditional manual techniques and CNN.

7.1 Assessing Manual Feature Extraction Techniques

This section presents four manual feature extraction techniques (HOG, SIFT, GLCM, and Harris Corners).

Histogram of Gradients (HOG)

We used the following code block to extract the HOG features from the five images. A 4x4 and 8x8 pixels per cell were used to assess HOG feature extraction.

```
from skimage.feature import hog
fd_4x4, hog_img_4x4 = hog(image, orientations=9, pixels_per_cell=(4, 4),
                    cells_per_block=(2, 2), visualize=True, channel_axis = -1)
fd_8x8, hog_img_8x8 = hog(image, orientations=9, pixels_per_cell=(8, 8),
                   cells_per_block=(2, 2), visualize=True, channel_axis = -1)
```

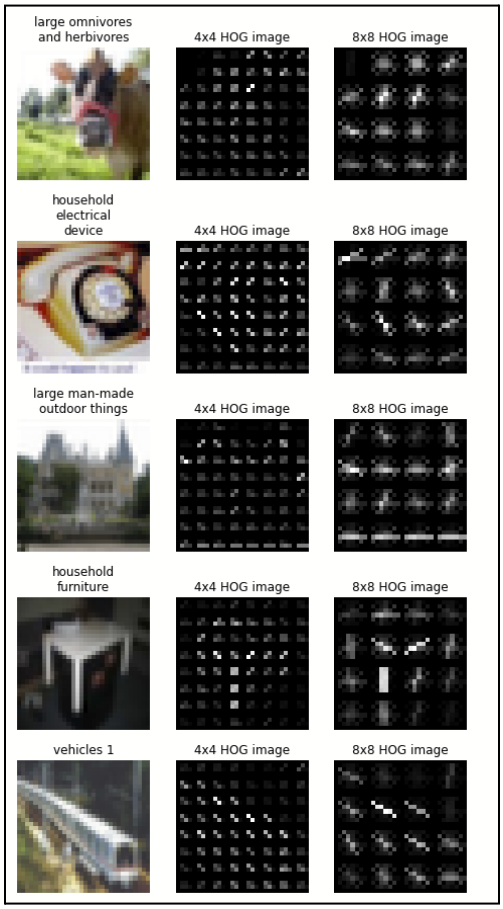


Figure 7.1: HOG Features for Sample Images

The HOG descriptor has a few key advantages over other descriptors. Since it operates on local cells, it is invariant to geometric and photometric transformations, except for object orientation.

Scale-invariant feature transform (SIFT)

SIFT is a popular shape feature extraction algorithm. It was proposed by Lowe (2004) to extract key interesting points.

We used the following code block to extract the SIFT features from the five images. We use the OpenCV SIFT detection object to compute features and descriptors of the grayscale image. Here we use nfeatures = 20 as the hyper-parameter. OpenCV uses DoG (Difference of Gaussians) to detect key points and then computes a feature vector for the surrounding region of each key point.

```
from google.colab.patches import cv2_imshow

size = 200

# Image 1
img = image_list[0][1]
title = image_list[0][0]

sift = cv2.SIFT_create(nfeatures=20)
kp = sift.detect(img,None)
kp, des = sift.compute(img, kp)

cv2.drawKeypoints(img, kp, img_sift, (51, 163, 236),
                  cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

resized_img_sift = cv2.resize(img_sift, (size,size))
black = [0,0,0]     #---Color of the border---
img1_border = cv2.copyMakeBorder(resized_img_sift,10,10,10,10,cv2.BORDER_CONSTANT,value=black )

# Display 5 Images
hcat = cv2.hconcat((img1_border,img2_border,img3_border,img4_border,img5_border))
cv2_imshow(hcat)
```

Figure 7.2: SIFT Features for Sample Images

SIFT has many advantages, such as invariance to illumination changes, robustness towards scaling, and orientation of the images.

Gray Level Co-occurrence Matrices (GLCMs)

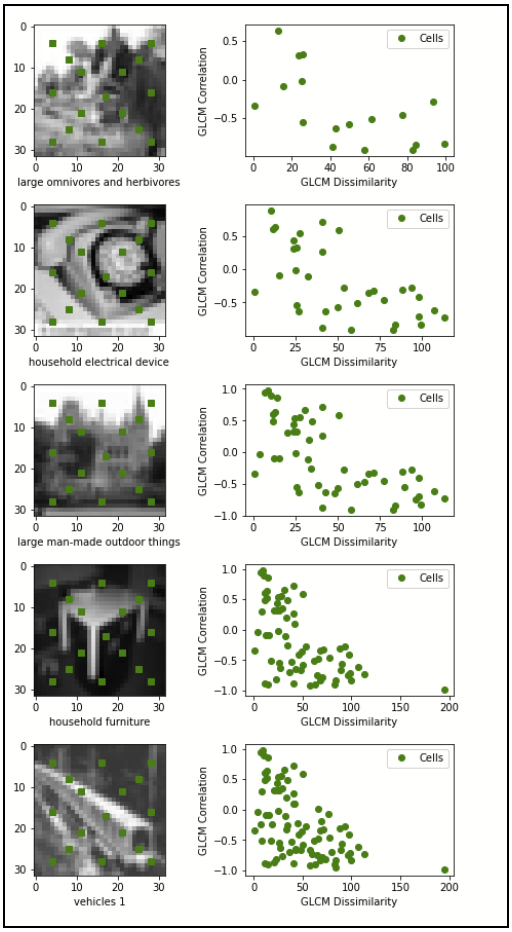
We used the following code block to extract the GLCM features from the five images. For each image, patch locations are specified along with Patch Size to derive the GLCM features.

```
from skimage.feature import graycomatrix, graycoprops
PATCH_SIZE = 6
# Define Patch Location to Generate GLCM Features
cell_locations =
[(1,1),(25,25),(1,25),(13,25),(25,13),(25,1),(1,13),(13,1),(5,5),(22,5),(5,22),(22,22),(14,14),(8,8
),(18,18),(18,8),(8,18)]
cell_patches = []
for item in image_list:
  img = item[1]
                                        img_color = item[2]
  title = item[0]
  # img = image_list[1][1]
  for loc in cell_locations:
    cell_patches.append(img[loc[0]:loc[0] + PATCH_SIZE, loc[1]:loc[1] + PATCH_SIZE])

# compute GLCM Properties for Each Patch
  diss_sim = []
  corr = []
  homogen = []
  energy = []
  contrast = []
  for patch in (cell_patches):
      glcm = graycomatrix(patch, distances=[5], angles=[0], levels=256,
                          symmetric=True, normed=True)
      diss_sim.append(graycoprops(glcm, 'dissimilarity')[0, 0])
      corr.append(graycoprops(glcm, 'correlation')[0, 0])
      homogen.append(graycoprops(glcm, 'homogeneity')[0, 0])
      energy.append(graycoprops(glcm, 'energy')[0, 0])
      contrast.append(graycoprops(glcm, 'contrast')[0, 0])
```

Figure 7.3: GLCM Features for Sample Images

In the below picture, the original grayscale image on the left with the cell patch locations is designated as green dots. On the right is a scatter plot of GLCM dissimilarity with GLCM correlation. Images with large texture differences will yield high dissimilarity and low correlation values.



Harris Corners Detection

We used the following code block to extract the Harris corner features from the five images. We use the OpenCV function cv2.cornerHarris to detect corners in an image. We use a grayscale image as the input image. We use a Sobel operator value of 23 so only the corners are detected.

```
img_gray = image_list[0][1]
title = image_list[0][0]
img_float = img_gray.astype(np.float32) / 255.0
dst = cv2.cornerHarris(img_float, 2, 23, 0.04)
img_color[dst > 0.01 * dst.max()] = [255, 0, 0]
ax[0].imshow(img_color)
ax[0].set_title(f"{title}", fontsize="9")
ax[0].axis('off')
```

Figure 7.4: Harris Corner Features for Sample Images

7.2 Assessing Learned Feature Extraction Techniques Through Neural Networks

Refer to Sections 9.1 and 9.2 for a description of learned features for Vision Transformers (ViT) and Convolutional Neural Networks (CNN).

8. Methodology and Evaluation Metrics

In this section, we describe our methodology and modeling approach. We compare three approaches for image recognition and image classification.

- *Method 1: Use Manual Feature Extraction Techniques (HOG, SIFT, and GLCM) and Apply a Classifier* (SVM, Random Forest, Logistic Regression)
    - Dataset: CIFAR-100 with 100, 20 Classes, and 5 Classes
    - Dataset: GLDv2 with 20 classes

- *Method 2: Use CNN Architecture (ResNet 9 and ResNet 50) for Feature Extraction and Classification*
    - Dataset: CIFAR-100 with 100 Classes

- *Method 3: Use Pre-Trained Vision Transformers (ViT) for Feature Extraction and Classification*
    - Dataset: CIFAR-100 with 20 Classes and 100 Classes
        - ViT Pre-Trained Only
        - ViT Pre-Trained and Fine Tuned
    - Dataset: GLDv2 with 20 classes
        - ViT Pre-Trained and Fine Tuned

It is our hypothesis that while CNN has been the state-of-the-art for image recognition and classification, Vision Transformers, trained on a vast corpus of images and fine-tuned, can also offer a superior solution.

The following evaluation metrics will be used for model comparison and evaluation.

Method 1: Accuracy
Method 2: Accuracy, Precision, Recall, F1-Score
Method 3: Accuracy, Precision, Recall, F1-Score, and Confusion Matrix

9.1 Vision Transformer Architecture

Image classification is a fundamental task in computer vision that involves assigning a label to an image based on its content. Vision Transformers (ViT) is a competitive alternative to Convolutional Neural Networks (CNNs) that are currently state-of-the-art in different image recognition computer vision tasks. Just as transformers-based models have revolutionized NLP, we're now seeing an explosion of papers applying them to all sorts of other domains.

Vision Transformer (ViT) achieves remarkable results compared to convolutional neural networks (CNN) while obtaining substantially fewer computational resources for pre-training. The ViT model represents an input image as a series of image patches, like the series of word embeddings used when using transformers to text, and directly predicts class labels for the image.

CNN uses pixel arrays, whereas ViT splits the input images into visual tokens. The visual transformer divides an image into fixed-size patches, correctly embeds each of them, and includes positional embedding as an input to the transformer encoder. Moreover, ViT models outperform CNNs by almost four times when it comes to computational efficiency and accuracy. The self-attention layer in ViT makes it possible to embed information globally across the overall image. The model also learns from training data to encode the relative location of the image patches to reconstruct the structure of the image.

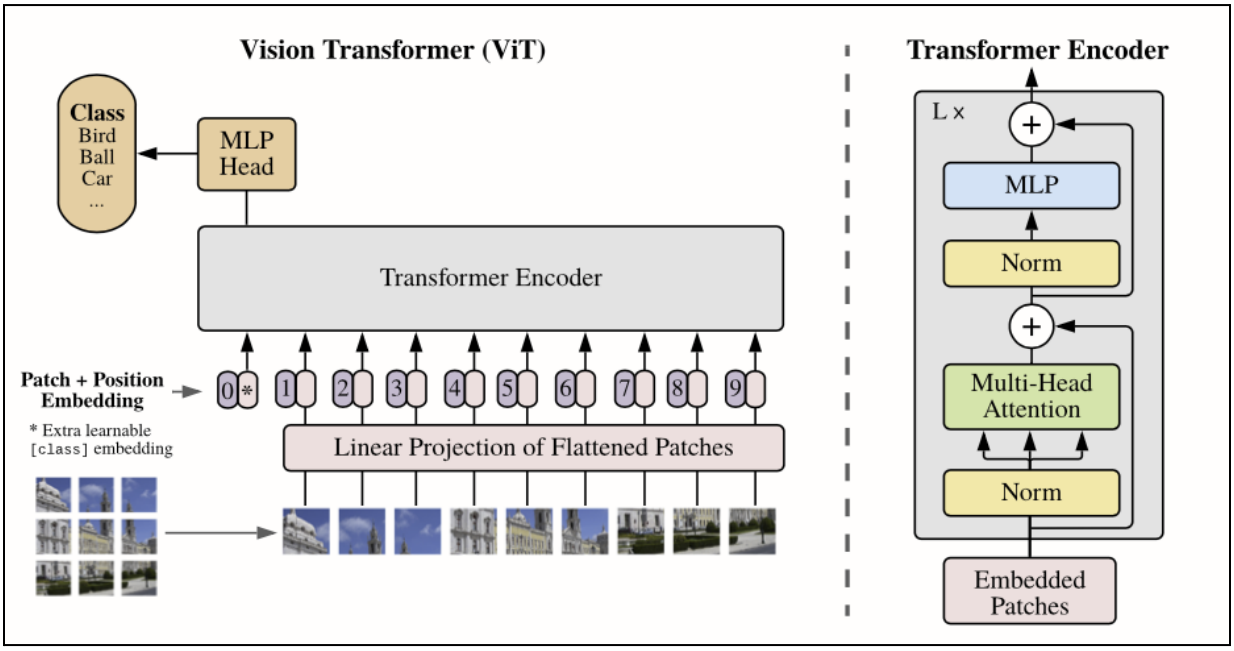


Figure 9.1: Vision Transformer Architecture

The overall structure of the vision transformer architecture consists of the following steps:

1. Break the image into patches (fixed sizes)
2. Flatten the patches
3. Generate lower-dimensional linear embeddings from the patches
4. Integrate positional embeddings
5. Feed the sequence to the transformer encoder
6. Pre-train the ViT model with image labels
7. Fine-tune the downstream dataset for image classification

Code Block for ViT using Hugging Face API

```
from transformers import ViTFeatureExtractor

# import model
model_id = 'google/vit-base-patch16-224-in21k'
feature_extractor = ViTFeatureExtractor.from_pretrained(
    model_id
)

def preprocess(batch):
    # take a list of PIL images and turn them to pixel values
    inputs = feature_extractor(
        batch['img'],
        return_tensors='pt'
    )

    # include the labels
    inputs['coarse_label'] = batch['coarse_label']
    return inputs

# transform the training dataset
prepared_train = dataset_train.with_transform(preprocess)

# ... and the testing dataset
prepared_test = dataset_test.with_transform(preprocess)

def collate_fn(batch):
    return {
        'pixel_values': torch.stack([x['pixel_values'] for x in batch]),
        'labels': torch.tensor([x['coarse_label'] for x in batch])
    }

from datasets import load_metric

# accuracy metric
metric = load_metric("accuracy")
def compute_metrics(p):
    return metric.compute(
        predictions=np.argmax(p.predictions, axis=1),
        references=p.label_ids
    )
```

9.2 Convolution Neural Network Architecture

Many Convolutional Neural Networks (CNNs) have been developed for image classification tasks. Here are some of the most popular ones:

1- **LeNet:** One of the earliest CNNs, developed by Yann LeCun in the 1990s. It consisted of 7 layers and was designed for handwritten digit recognition.
2- **AlexNet:** A deep CNN that won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012. It consists of 8 layers, including 5 convolutional layers and 3 fully connected layers.
3- **VGGNet:** A series of CNNs developed by the Visual Geometry Group at the University of Oxford. VGG16 and VGG19 are the most popular variants, consisting of 16 and 19 layers, respectively.

4- **GoogLeNet** (Inception): A CNN developed by Google that won the ILSVRC in 2014. It uses a unique "Inception module" that efficiently uses parameters and enables the network to be deeper and broader.
5- **ResNet:** A CNN developed by Microsoft Research that won the ILSVRC in 2015. It uses residual connections to enable the training of much deeper networks with hundreds or even thousands of layers.
6- **DenseNet:** A CNN that connects each layer to every other layer in a feed-forward fashion. It has very high parameter efficiency and has achieved state-of-the-art results on a number of image classification tasks.
7- **EfficientNet:** A family of CNNs that use a compound scaling method to balance model accuracy and computational efficiency. EfficientNet has achieved state-of-the-art performance on ImageNet with significantly fewer parameters than other models.

These are just a few examples of the many CNNs that have been developed for image classification tasks. Each CNN has its own strengths and weaknesses and may perform better on specific tasks and datasets.

For this project, we picked ResNet. ResNet (short for Residual Network) is a deep neural network architecture that was introduced in 2015 by researchers at Microsoft Research. The key innovation of ResNet is the introduction of residual connections, which allow for the training of much deeper networks with hundreds or even thousands of layers. Residual connections enable the training of deeper networks by addressing the vanishing gradient problem. This problem occurs when the gradients of the loss function with respect to the weights of earlier layers in a deep network become very small, making it difficult to update these weights and slowing down the training process. Residual connections allow the gradient to flow more quickly through the network, improving training and enabling the use of much deeper networks.

ResNet has become one of the most popular architectures for deep learning tasks, such as image classification, object detection, and segmentation. The ResNet architecture has been extended and modified in various ways to improve its performance on specific tasks, such as the introduction of bottleneck layers to reduce the computational cost of training and the use of ResNeXt, a variant of ResNet that uses grouped convolutions to reduce the number of parameters in the network.

ResNet-9 and ResNet-50 are two different variants of the ResNet architecture that differ in their depth and number of layers. ResNet-9 is a smaller variant of ResNet that consists of only 9 layers, including one convolutional layer and 8 residual blocks. This makes ResNet-9 more shallow than ResNet-50. ResNet-9 is typically used for smaller datasets or when computational resources are limited. ResNet-50, on the other hand, is a much deeper and larger variant of ResNet that consists of 50 layers, including multiple convolutional layers and residual blocks. ResNet-50 is designed to handle more complex and larger datasets and can achieve state-of-the-art performance on many computer vision tasks. The main difference between ResNet-9 and ResNet-50 is their depth and the number of layers. ResNet-50 has many more layers than ResNet-9, which makes it more powerful but also more computationally expensive. Consequently, ResNet-50 may require more computational resources and longer training times than ResNet-9, but it can achieve higher accuracy on more complex tasks.

10. Experimental Results

Results for the three methods described in Section 8 - Methodology and Evaluation Metrics are shown in this section.

- **Table 10.1** provides an “Accuracy” comparison of manual feature extraction techniques with a classification layer applied for the CIFAR-100 dataset.
- **Table 10.2** provides accuracy, precision, recall, and confusion matrix for the CNN Model (ResNet 9 and ResNet 50) on CIFAR-100
- **Table 10.3** provides accuracy, precision, recall, and confusion matrix for the ViT Model on CIFAR-100
- **Table 10.4** provides an “Accuracy” comparison of manual feature extraction techniques and the ViT Model on the GLDv2 dataset.
- **Figure 10.1** provides the confusion matrix for the ViT model fine-tuned for CIFAR-100

Table 10.1: Method 1 Test Accuracy Results for CIFAR-100
(Use Manual Feature Extraction Techniques with a Classifier)

			Feature Extraction Technique (Accuracy)		
	Classifier	**Data Set**	**HOG**	**SIFT**	**GLCM**
20 Course Classes					
1	SVM	CIFAR 100	39.79%	16.74%	22.42%
2	Logistic Regression	CIFAR 100	30.19%	13.38%	13.39%
3	Random Forest	CIFAR 100	29.05%	12.18%	16.74%
5 Course Classes					
4	SVM	CIFAR 100	79.2%	37.08%	46.32%
5	Logistic Regression	CIFAR 100	73.8%	32.18%	33.28%
6	Random Forest	CIFAR 100	73.4%	34.22%	43.52%

Table 10.2: Method 2 Test Precision, Recall, F-1 Score, and Accuracy Results for CIFAR-100
(CNN)

	CNN Model	**Data Set**	**Accuracy**	**Precision**	**Recall**	**F-1 Score**
100 Fine Classes						
1	CNN-ResNet 9	CIFAR100 - 100 Classes	73.32%	73.56%	73.32%	73.32%
2	CNN-ResNet 50	CIFAR 100 - 100 Classes	73.86%	74.1%	73.86%	73.85%

Table 10.3: Method 3 Test Precision, Recall, F-1 Score and Accuracy Results for CIFAR-100 (Vision Transformer)

	ViT Model	Data Set	Accuracy	Precision	Recall	F-1 Score
20 Course Classes						
1.a	ViT Pretrained Only (google/vit-base-patch16-224-in21k)	CIFAR 100 - 20 Classes	8.57%	8.76%	8.57%	7.11%
1.b	ViT Pretrained + Fine Tuned (google/vit-base-patch16-224-in21k)	CIFAR 100 - 20 Classes	94.13%	94.15%	94.13%	94.13%
100 Fine Classes						
2.a	ViT Pretrained Only (google/vit-base-patch16-224-in21k)	CIFAR 100 - 100 Classes	1.64%	1.08%	1.64%	1.07%
2.b	ViT Pretrained + Fine Tuned (google/vit-base-patch16-224-in21k)	CIFAR 100 - 100 Classes	89.93%	90.11%	89.30%	89.95%

Table 10.4: Accuracy Results using GLDv2 dataset with HOG as Manual Feature Extraction and Pre-trained Vision Transformer

	Classifier	Data Set	Accuracy
20 Classes			
1	SVM	GLDv2	46.22%
2	Logistic Regression	GLDv2	37.93%
3	Random Forest	GLDv2	29.15%
4	ViT Pretrained + Fine Tuned (google/vit-base-patch16-224-in21k)	GLDv2	98.14%

Figure 10.1: Confusion Matrix for Vision Transformer + Fine Tuned for Coarse Labels for CIFAR-100 (Vision Transformer)

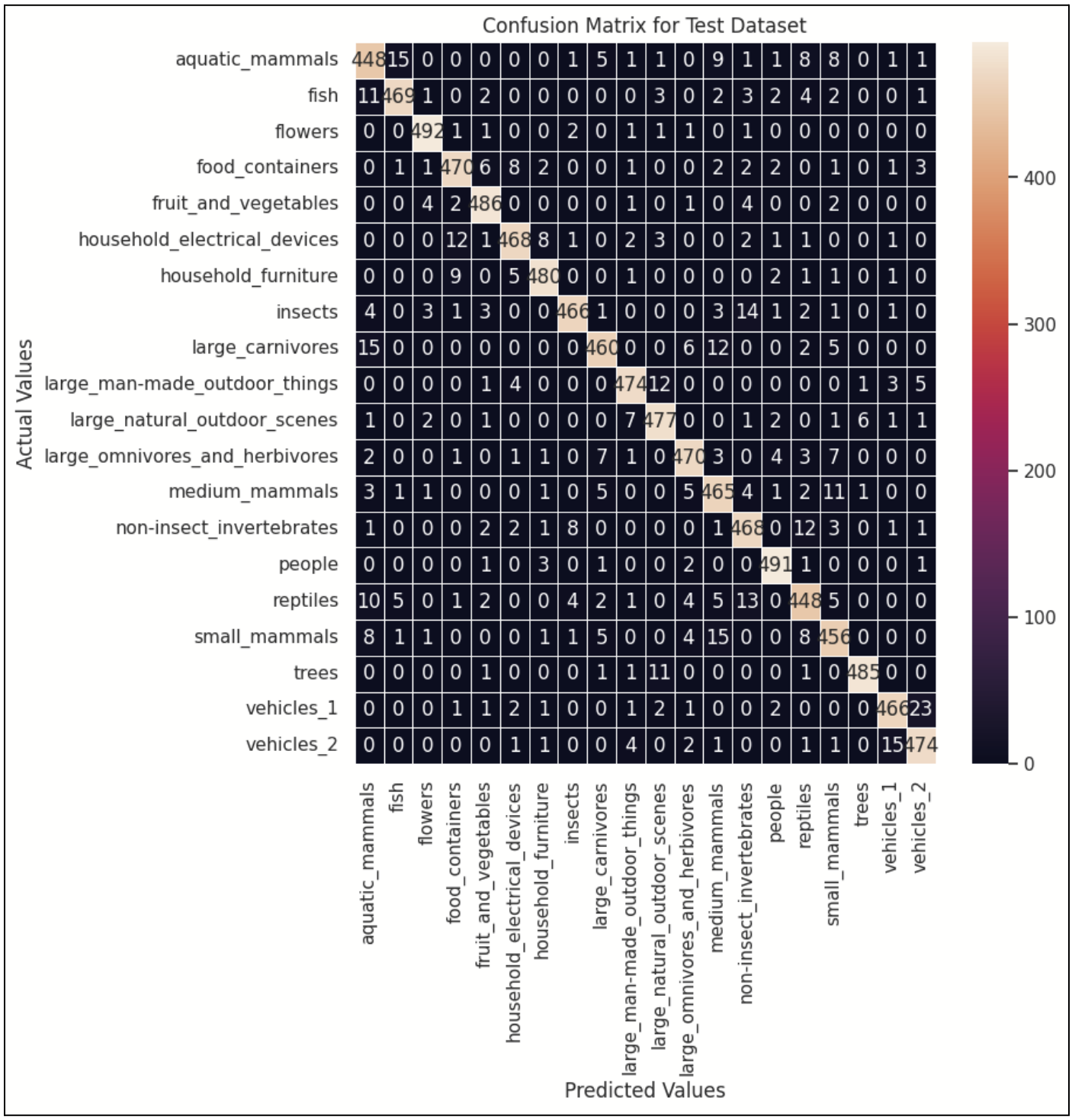


Figure 10.2: True and Predict Coarse Labels for Random Test Images from ViT + Fine Tuned

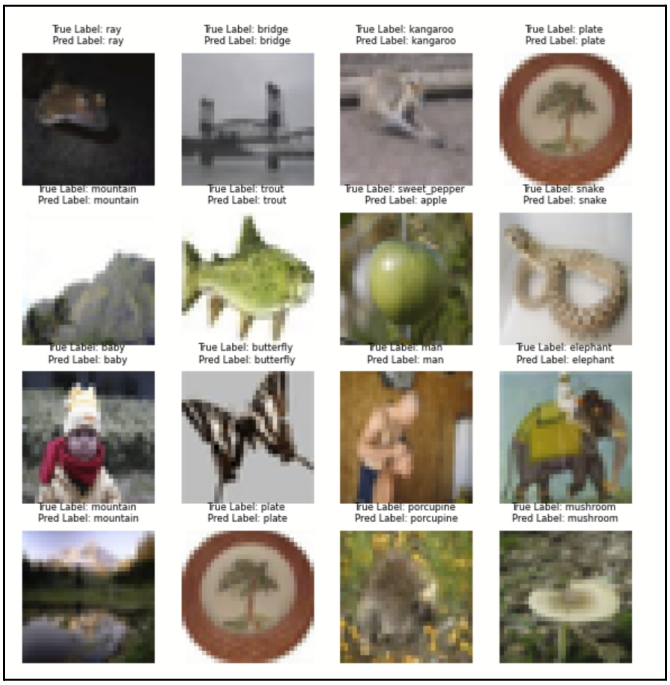


Figure 10.3: True and Predict Fine Labels for Random Test Images from ViT + Fine Tuned

11. Discussion and Analysis

From Section-10 Experimental Results, the following conclusions can be ascertained:

CIFAR-100

Manual Feature Extraction Techniques with a Classifier

- Table 10.1 shows that HOG+SVM is the best combination of feature extraction and classifier on CIFAR-100. HOG outperformed SIFT and GLCM as feature extractors.
- All the models performed poorly when the number of classes was 20, with the lowest accuracy for SIFT+Random Forest at 12.18% and the highest for HOG+SVM at 39.79%.
- This is not surprising given the differences in the features amongst the images, and no single feature extractor can work effectively across all the image classes.
- As the number of classes was reduced to 5, HOG+SVM achieved a 79.2% accuracy.

Pre-Trained Vision Transformer with Fine Tuning on CIFAR-100

- From Table 10.3, a pre-trained ViT model that was fine-tuned on CIFAR-100 performed remarkably well with an accuracy of 94.13% and an F1-Score of 94.13%.
- Even when the number of classes increased from 20 to 100, the model still performed well with an accuracy of 89.93% and an F1-Score of 89.95% (See Table 10.3).
- Figure 10.1 Confusion Matrix illustrates misclassifications of the model on the CIFAR-100 test data. Interestingly, the majority of misclassifications are in adjacent classes. As an example, 15 of the actual fish labels were misclassified as aquatic mammals. Twenty-three of the actual labels of vehicles_1 were misclassified as vehicles_2.
- Figures 10.2 and 10.3 provide a visual inspection of 16 random test images from CIFAR-100 with associated predictions for coarse labels and fine labels. In Figure 10.2, an image with a true label of flowers is predicted as a large natural outdoor. In Figure 10.3, an image with a true label of sweet pepper is predicted as an apple.
- Higher-resolution images and a more distinct class can help reduce misclassifications.

CNN Models-

- From Table 10.2, Best CNN performance on ResNet 50 but still inferior to Vision Transformer Model

GLDv2

Manual Feature Extraction and Pre-Trained Vision Transformer with Fine Tuning on GLDv2

- From Table 10.4, similar to CIFAR-100, the Pre-Trained Vision Transformer with Fine Tuning had much better accuracy than using Manual Feature Extraction models.
- Pre-Trained Vision Transformer performed exceptionally well with an accuracy of 98%, more than twice the accuracy of Handcrafted Models.
- The full-sized Kaggle GLDv2 dataset was more than 100GB, which presented a more significant computational challenge than working with the CIFAR-100 dataset. However, we suspect that having images with much better resolution helped improve the model accuracy since all models trained with the GLDv2 dataset had higher accuracy than the similar models built with CIFAR-100, even when the GLDV2 dataset contained fewer images per class.

12. Generalizability

From Section 11 - Discussion and Analysis, the Vision Transformer pre-trained and fine-tuned on CIFAR-100 generalizes very well on Test Dataset. With 20 course classes, the model F1-Score is 94.13% and has an accuracy of 94.13%; with 100 fine classes, the model F1-Score is 89.95%, with an accuracy of 89.93%.

What is highly remarkable is that the model performs so well with a 32 by 32 image resolution of the CIFAR-100 dataset. With higher-resolution images and distinct classes, the ViT should achieve even better state-of-the-art results. This is clear from Table 10.4 in Section 10, where the test accuracy on GLDv2 was 98.14%.

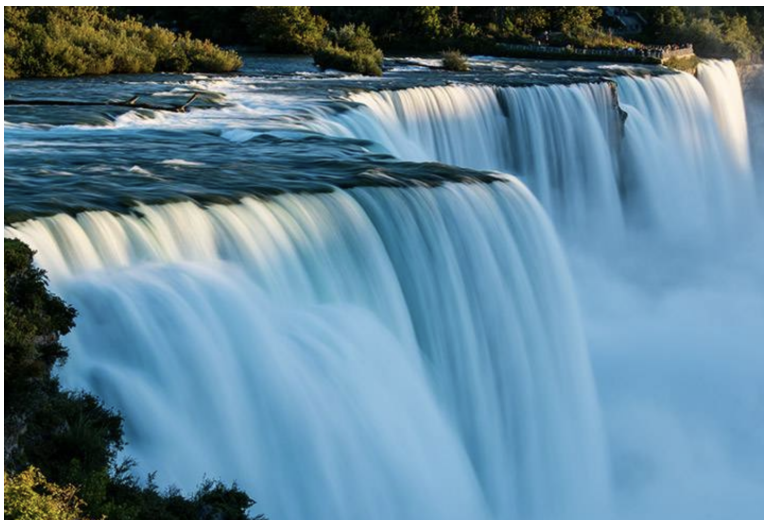
Table 12.1 provides an accuracy assessment by coarse class for CIFAR-100. Classes that are distinct and have good features have higher accuracy rates. For example, flowers, people, trees, fruits, and vegetables have the highest accuracy rates. Aquatic mammals and reptiles have the lowest accuracy rates. These classes are not distinctive and have other classes, such as small mammals and medium mammals, that are close.

Table 12.1: Vision Transformer Test Accuracy by Coarse Classes for CIFAR-100

Coarse Class	Accuracy	Coarse Class	Accuracy	Coarse Class	Accuracy
aquatic_mammals	**89.6%**	insects	93.2%	people	**98.2%**
fish	93.8%	large_carnivores	92.0%	reptiles	**89.6%**
flowers	**98.4%**	large_man-made_outdoor_things	94.8%	small_mammals	91.2%
food_containers	94.0%	large_natural_outdoor_scenes	95.4%	trees	**97.0%**
fruit_and_vegetables	**97.2%**	large_omnivores_and_herbivores	94.0%	vehicles_1	93.2%
household_electrical_devices	93.6%	medium_mammals	93.0%	vehicles_2	94.8%
household_furniture	**96.0%**	non-insect_invertebrates	93.6%		

To test the model's generalizability outside the GLDv2 dataset, we tried the model with random images from the internet, images completely outside the training dataset, and the model classified all of them correctly.

To further test if the model would generalize to unknown images, we tried the model with drawings of the landmarks or even sketches with low details. Still, the model correctly classified the drawings and sketches.

Random internet image Real class: Niagara_Falls Predicted class: Niagara_Falls	
Random internet drawing Real class: Golden Gate Predicted class: Golden Gate	

Random internet sketch Real class: Golden Gate Predicted class: Golden Gate	
Random internet sketch Real class: Eiffel Tower Predicted class: Eiffel Tower	

13. Efficiency versus Accuracy

In this section, we discuss the trade-off between computational resource requirements and accuracy.

For the CIFAR-100 dataset, feature extraction with traditional CV methods like HOG, SIFT, and GLCM and applying the classification layer like SVM, Random Forest, and Logistic Regression required significantly less computation time and compute resources.

CNN and ViT Models required considerably more computing resources and training time. The size of the images also plays a significant role in run time. See Table 13.2 for GLDv2 Train Run Time.

Table 13.1: Train Run Time Comparison for Deep Learning Models for CIFAR-100

Model	# of Training Samples	# of Classes	Train Run Time
ViT (Pre-Trained+Fine Tuned)	50,000	20	6 hours 47 min
ViT (Pre-Trained+Fine Tuned)	50,000	100	6 hours 38 min
CNN ResNet 9 [No pre-trained model]	50,000	100	1 hour 10 min
CNN ResNet 50 [No pre-trained model]	50,000	100	1 hour 50 min

Table 13.2: Train Run Time Comparison for Deep Learning Models for GLDv2

Model	# of Training Samples	# of Classes	Train Run Time	Compute resource
ViT (Pre-Trained+Fine Tuned)	16,633	20	11 hours 6 min with, 8Cores and 30GB RAM	8 Cores 30 GB RAM Nvidia A100 GPU
HOG with SVM, Random Forest, and Logistic Regression	16,633	20	2 Hours 53 min	8 Cores 30 GB RAM

14. Conclusion and Next Steps

To conclude, based on our experiments using manual feature extraction techniques, CNN models, and Vision Transformers, it is evident that a pre-trained Vision Transformers that is fine-tuned achieves outstanding classification accuracy across large, complex datasets even when the numbers of classes are large.

While traditional techniques work well on small and less complex datasets, Vision Transformers scale well and achieve state-of-the-results for a wide range of applications from simple to complex. Pre-training and fine-tuning ViT are essential, as just the pre-trained model without fine-tuning achieves mediocre results.

Future studies can include experiments to understand under what conditions a ViT may underperform, say CNN, such as heterogeneous images containing many objects. Also, combining features extracted by a ViT with handcrafted techniques can address some intra-class variation and inter-class similarities, especially with medical imaging. Using multi-modal transformer models with NLP and CV can lead to compelling use cases. For example, classifying and understanding medical images and generating a technical report can be a very useful application in the medical imaging industry.

15. References

1. Gidudu Anthony, Hulley Greg, Marwala Tshilidzi. **Classification of Images Using Support Vector Machines**. *Department of Electrical Engineering and Information Engineering, University of Witwatersrand, South Africa.*
2. Jaya Dwan, Sudeep Thepade, **Image Retrieval Using Low Level and Local Features Contents: A Comprehensive Review**. *Applied Computational Intelligence and Soft Computing. 2020.*
3. Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unthethiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jacob Uszkoreit, Neil Houlsby. **An Image is worth 16x16 words: Transformers for Image Recognition at Scale**. *Published at ICLR 2021.*
4. Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, Mubarak Shah. **Transformers in Vision: A Survey**. *arXiv:2101.01169v5. 2022.*
5. Afshan Latif, Aqsa Rasheed, Umer Sajid, Jameel Ahmed, Nouman Ali, Naeem Iqbal Ratyal, Bushra Zafar, Saadat Hanif Dar, Muhammad Sajid, Tehmina Khalil. **Content-Based Image Retrieval and Feature Extraction: A Comprehensive Review**. Hindwi, *Mathematical Problems in Engineering. 2019.*
6. David Lowe. **Distinctive Image Features from Scale-Invariant Keypoints**. *International Journal of Computer Vision. 2004*
7. Hui Hui Wang, Dzulkifli Mohamad, N.A. Ismail. **Image Retrieval: Techniques, Challenge and Trend**. *World Academy of Science, Engineering and Technology. International Journal of Computer, Electrical, Automation, Control and Information Engineering. 2009.*
8. Tobias Weyand, Andre Araujo, Bingyi Cao, Jack Sim, Google Research. **Google Landmarks Dataset v2**. *arXiv:2004.01804v2. 2020.*