/

DRAFT

/

Natural Images Classification Project Report

Ayelet Katkov Hila Shamir

January 2025

Deep Learning and Natural Languages Processing Course, Ariel University

Abstract

This is a report on our project, of classifying images from the “Natural Images” dataset to their correct label out of the eight options.

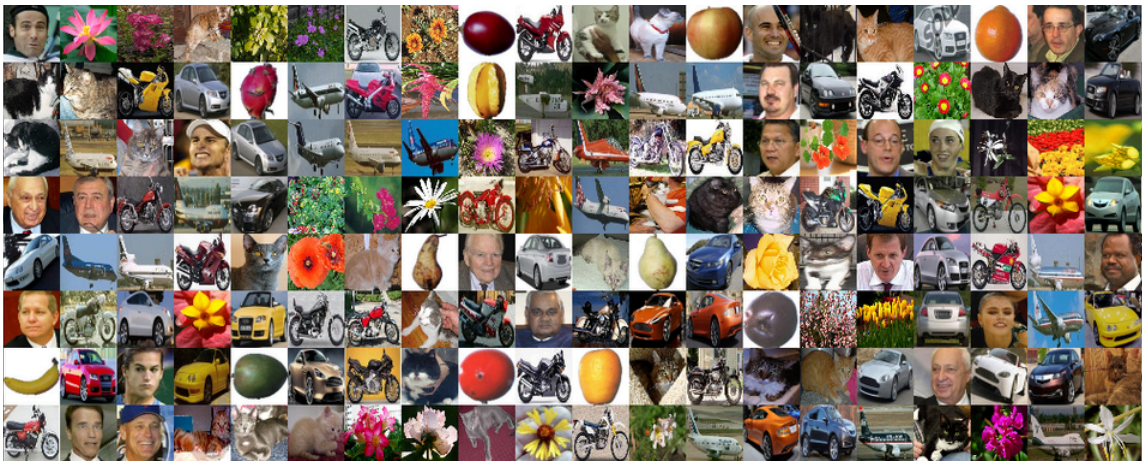
Along the report several attempts will be presented, with graduating difficulty level, aiming to improve accuracy of the predicting models.

We will present a baseline “dummy” classifier, softmax model, simple fully-connected neural network, simply convolutional neural network and some advanced ones. At last we will share results of using pytorch’s pre-trained neural network of type VGG16.

Dataset introduction

The dataset we will be focusing on during this report is as mentioned the “Natural Images” dataset, containing 6899 images from 8 classes:

[airplane, car, cat, dog, flower, fruit, motorbike, person].



Images are matrices of pixels. The matrices have two dimensions containing integer values in the range [0, 255] that describe the intensity of the color at that exact location, aka pixel. If the image is colored, a third dimension is added, describing the color combination based on proportions of some format, for example RGB - red green blue.

In the following classification models the features and the labels are as follows:

Features - are the pixel’s intensities, over the three color channels of RGB.

Labels - are the eight classes we map the images to.

Dataset preparation, preprocessing

תמונה	
מזהה תמונה	
מימדים	290 x 94
רוחב	290 פיקסלים
גובה	94 פיקסלים

תמונה	
מזהה תמונה	
מימדים	300 x 68
רוחב	300 פיקסלים
גובה	68 פיקסלים

תמונה	
מזהה תמונה	
מימדים	100 x 100
רוחב	100 פיקסלים
גובה	100 פיקסלים

תמונה	
מזהה תמונה	
מימדים	273 x 423
רוחב	273 פיקסלים
גובה	423 פיקסלים

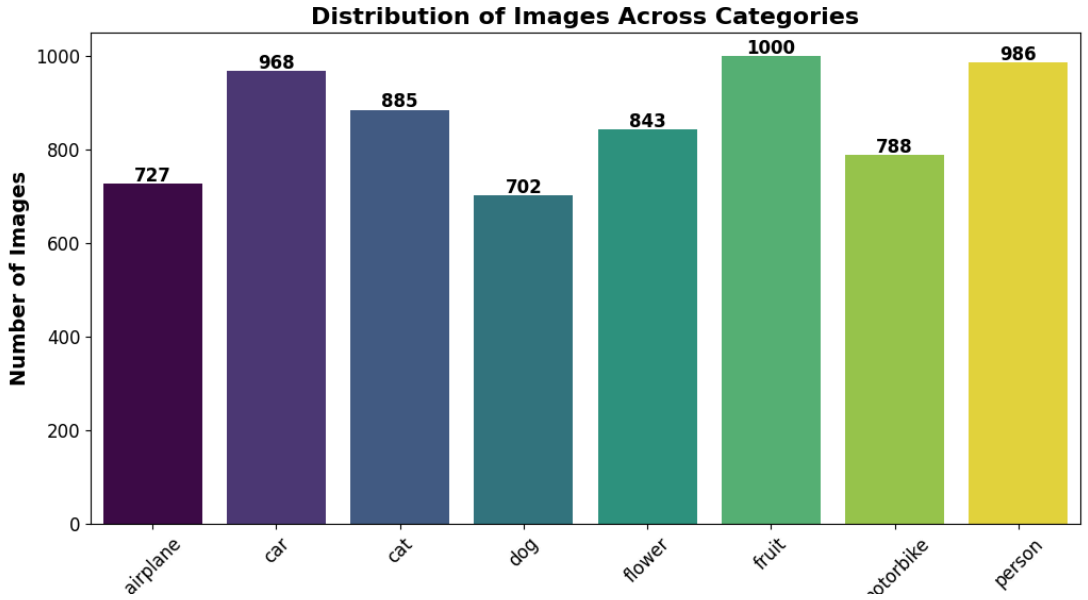
Since in this dataset images have different sizes, in order to send to the neural networks models uniform sized samples, preprocessing on the data will be required.

So as we load the images they are resized to (128*128) each.
Also before being used in neural networks all the images are normalized so that the values of all pixel's intensities transform from the range [0, 255] to have values in between [0, 1], by dividing all values by 255.
Before some of the the more complicated models, in order to increase the dataset for the model to have more samples to train on data augmentation will take place, composing images with torchvision's transforms.

Data Splitting

During the trials we will hold a consistent splitting percentage, training the different models on 80% of the dataset, and devote the remaining 20% to testing. In some cases the test's data split will be parted in two, leaving 10% for validation and 10% for test.

Performance Evaluation



Since the data distribution is nearly balanced, and since all types of misclassification have the same weight to us, in order to evaluate and compare the different models' results we will compare predictions' accuracy metric.

The accuracy is calculated by accuracy = $\frac{num.\,correctly\,classified}{all\,classified}$.

Baseline Predictor

For the baseline a simple "dummy" predictor is used.

The classifier will simply predict the most frequent class for all samples.
The implementation in the code uses “DummyClassifier” from sklearn library with the “most_frequent” predictions mode.
Below the results of the DummyClassifier test evaluation are shown.

	precision	recall	f1-score	support
0	0.00	0.00	0.00	161
1	0.00	0.00	0.00	201
2	0.00	0.00	0.00	170
3	0.00	0.00	0.00	132
4	0.00	0.00	0.00	170
5	0.13	1.00	0.23	183
6	0.00	0.00	0.00	151
7	0.00	0.00	0.00	212
accuracy			0.13	1380
macro avg	0.02	0.12	0.03	1380
weighted avg	0.02	0.13	0.03	1380

As we can see from the data distribution plot, most samples belong to the sixth class, “fruit”, so the classifier will predict this class to all of the samples.

Multi-Class Logistic Regression Classifier

Next presented attempt uses the LogisticRegression model from sklearn library, and compares results of the two multi-class handling approaches.
“ovr” - one-vs-rest. This model fits a separate binary classifier for each class K in the dataset. It computes the probability for each class from its respective classifier, where the final class is the one with the highest probability. $\hat{y} = argmax_k P(y = k|x)$

“multinomial” - softmax. This approach uses the softmax function to compute probabilities for all classes:

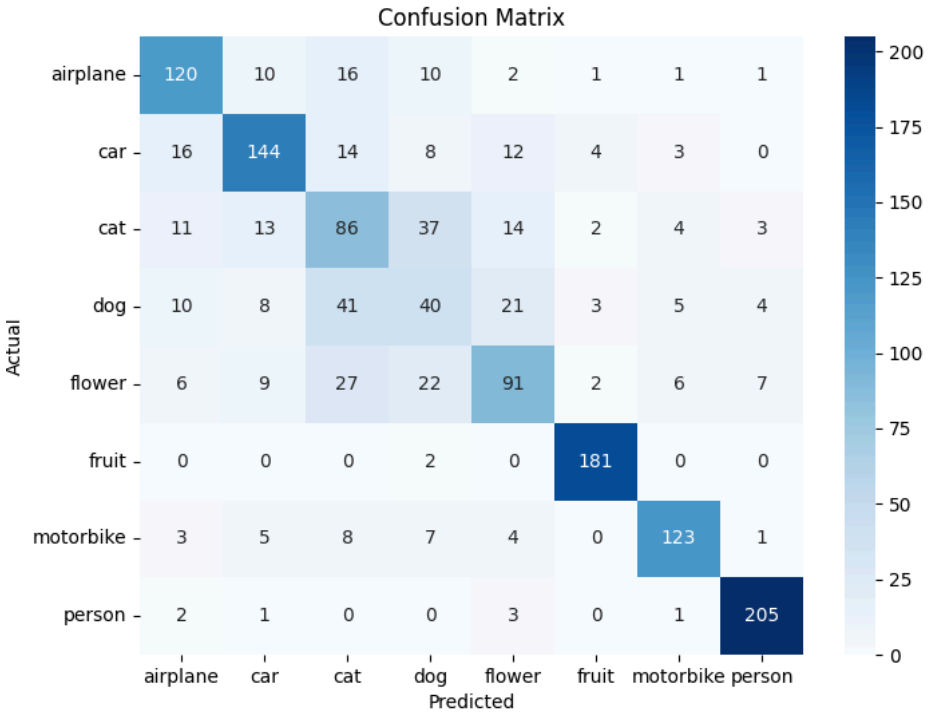
P(y=k|x) = $\frac{{w_k}^T * x + b_k}{\sum_{j=1}^{K} e^{{w_j}^T * x + b_j}}$ where w_k and b_k are the weight and bias for class k.

The model is trained by minimizing the cross-entropy loss, comparing the predicted probability distribution (from Softmax) with the true label (which is represented as a one-hot encoded vector) and penalizes the network if the predicted probability for the correct class is low.

Below is shown a table comparing the performance of both models, with maximum 1000 iterations (to give the optimization algorithm enough time to converge), and random state of 42 (to make the random choices of first weights consistent across all runs).

Approach Used	Achieved Accuracy
one-vs-rest	0.69
multinomial - softmax	0.717

As we can see, for the natural images dataset softmax showed a better performance.
The following confusion matrix shows the distribution of softmax's results across all classes:



Now let's see the difference in the achievements of the baseline model and the softmax according to the accuracy metric..

Model	Accuracy
baseline - most frequent predictor	0.13
multinomial - softmax	0.717

We can see that while using just a slightly more complicated model, drastic leap can already be seen.

FCNN - Fully Connected Neural Network

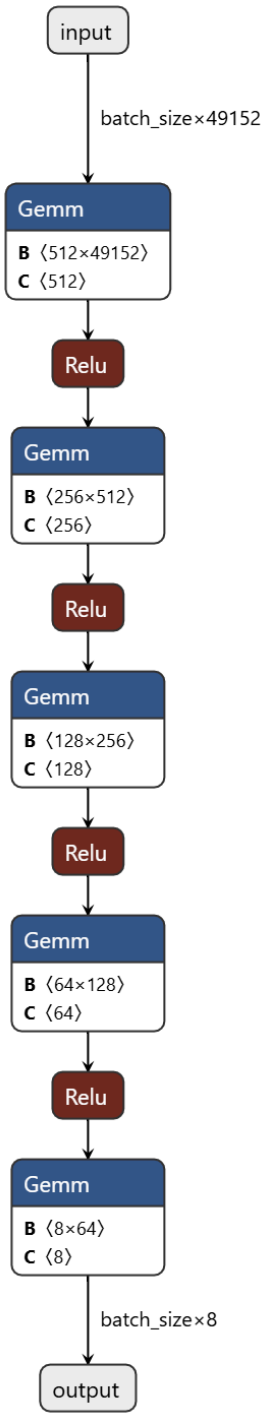
In this next attempt we will try to improve the results even more, using a simple feedforward fully connected neural network, and some experiments for improving it's results with some complications.

First of all, before using a neural network we normalized the values of all pixels from the range [0, 255] to have values in between [0, 1], by dividing all values by 255.

Then we will create

Then load the dastasets to the batches.

Note that several attempts with multiple hyper-parameters combinations and tunings have been made, this report will include some of them to show the tendency of improvement.



Model Structure: The simple fully connected neural network model is constructed from an input layer, three hidden layers and an output layer predicting the classes classification predictions.

See figure to the right ->.

With vs. without validation set split:

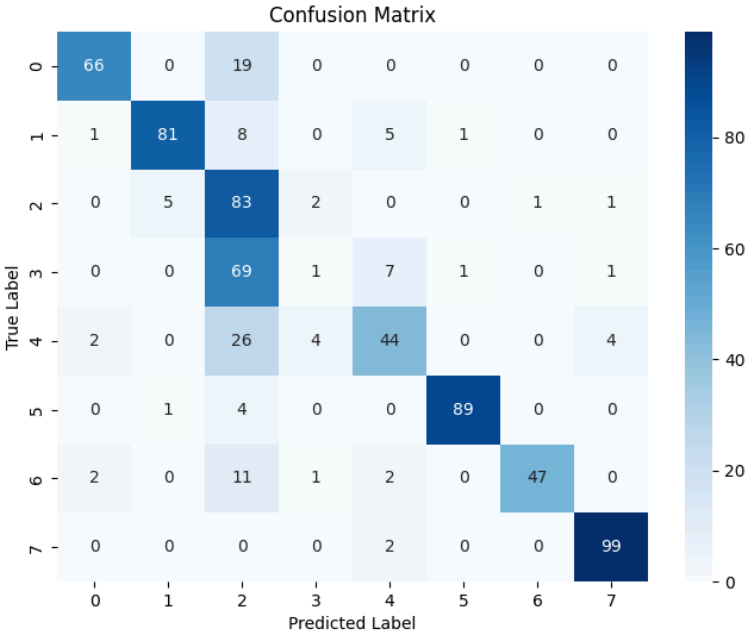
One of the model improvements we tried was devoting a part of the dataset to validation, and after each training loop evaluating the model on a batch of the validation samples, results are shown below (batch size = 32 and learning rate = 0.001 (hyper parameters choice that reached best accuracy)):

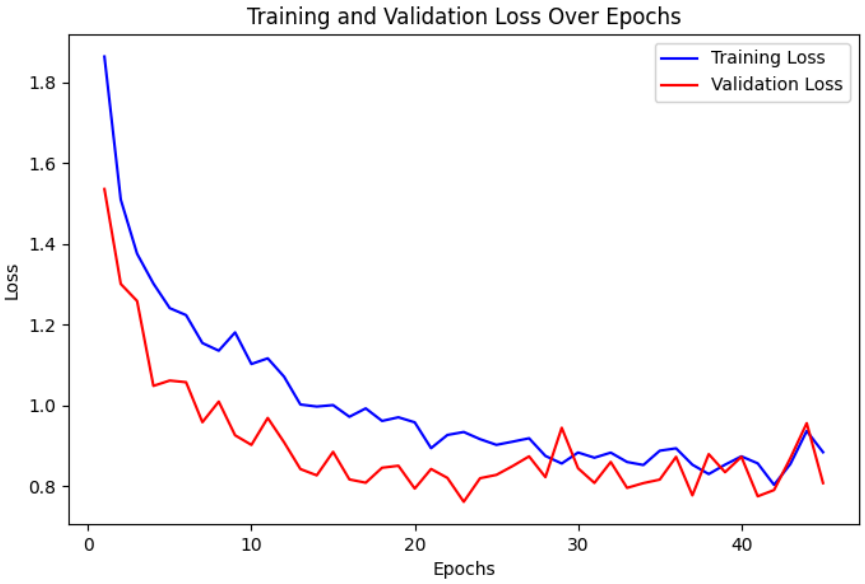
Model	20 epochs	35 epochs	45 epochs
FCNN 80% train 20% test	0.71	0.787	0.79
FCNN 80% train, 10% validation, 10% test	0.77	0.79	0.80

Since with more than 45 epochs the accuracy converged to these numbers, we concluded on the "natural images" dataset it's better to run the FCNN model on 45 epochs, and based on accuracy metric comparison using a validation set improved the predictions accuracy.

Early Stopping - The validation computation helps perform early stopping, tracking the validation loss tendency, and if for over 5 epochs the validation loss has only gotten worse then the model training is stopped.

The confusion matrix below describes the outcomes of the FCNN model, with validation set split, and with 45 epochs (best achieving combination based on our observations).





The figure above visualizes changes in training loss and validation loss along the training process of the FCNN model.

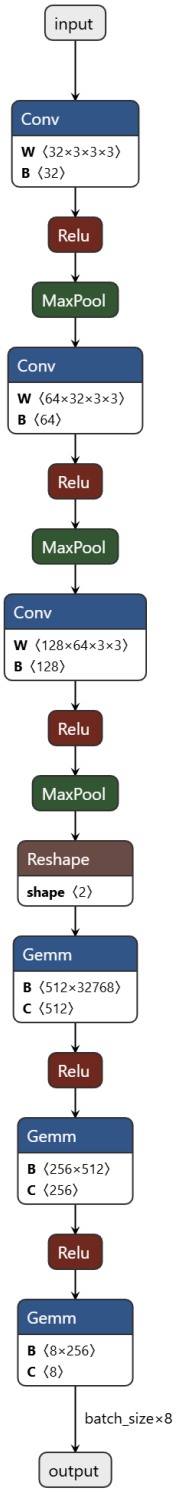
FCNN vs. previous models:

Model	Accuracy
baseline - most frequent predictor	0.13
multinomial - softmax	0.717
FCNN - simple feedforward fully connected neural network	0.80

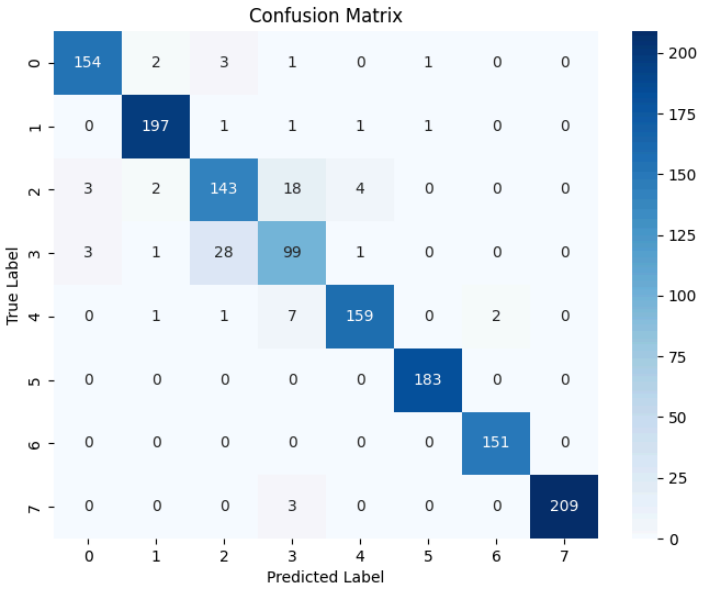
CNN - Convolutional Neural Network Model

Since the feedforward fully connected neural network takes a lot of resources, especially time, to train even only 5 layers, and the specialty of neural networks is combining many layers on top of each other to improve performances, if we have limited time to train the model we can not create many layers in a FCNN. So our next attempt will use CNN, a convolutional neural network.

The network will contain a features part (convolutional layers) and a classifier part (linear fully connected final layers that create the final prediction to classes). The main layers will be padded with RELU layers (converting any negative values to 0), max pooling and batch normalization layers.
In order to compute predictions loss cross entropy will be used, and for minimizing the loss function ADAM optimizer is used (adaptive moment estimation).
See our cnn structure schema to the right ->
An additional experiment of adding a validation set and comparing model's performances is described in the table below (batch size = 32 and learning rate = 0.001 (hyper parameters choice that reached best accuracy) equal along all different runs):



Model	20 epochs	35 epochs
CNN: 80% train 20% test	0.948	0.95
CNN: 80% train, 10% validation, 10% test	0.932	0.939



The figure above shows a confusion matrix of the CNN model's predictions.

Finally, let us compare the results of a CNN on this dataset vs. previous models.

Model	**Accuracy**
baseline - most frequent predictor	0.13
multinomial - softmax	0.717
FCNN - simple feedforward fully connected neural network	0.80
CNN - convolutional neural network	0.948

VGG16

One can notice the best score reached throughout this report was ~95% accuracy on the test samples.
Since the images processing domain and computer vision in general nowadays are very advanced, and many great neural networks and models have been made specifically to handle image datasets and return almost perfect predictions, the 95% on this way not too complicated dataset is clearly not the limit, so we wanted to compare the results of previously shown neural networks to existing, pre-trained on bigger datasets model's achievements.
Using a VGG16 network alongside with data preparation and augmentation impressively raised the accuracy on the test data up to 99.

Further work can be done in trying out other pre-trained neural networks.