Işık Özsoy	21703160	
Utku Kalkanlı	21602325	
Yahya Mahmoud Ahmed Elnouby Mohamed	21801332	
Mohammed Sameer Yaseen	21801331	Section: 2
Oğuz Orhun Tüzgen	21802313	TA: İlayda Beyreli

CS 464 - INTRODUCTION TO MACHINE LEARNING: Final Report
Computer Vision-Based Garbage Classification

1. Introduction

Currently, garbage is classified into recycling categories manually. However, performing that process manually is time and energy-consuming as well as economically inefficient. Given that the process is a repetitive one and is based on object detection and identification, we thought it would be more time and economy-efficient if that process was done automatically. The aim of the project is to find an efficient way to automate classifying trash. Therefore, a computer vision approach is used to provide predictions regarding objects' materials based on the images in our training datasets and their labels. We predicted the category of an object and classified it as a battery, organic material, glass (brown, green, or white), plastic, metal, cardboard, paper, or trash using Garbage Classification dataset [1,2].

The project is based on garbage classification by using transfer learning which focuses on using the knowledge learnt from the previous problems and applying it to other similar problems. In our project, the knowledge gained while learning to detect plastics could apply when trying to recognize papers. Another benefit of transfer learning is the decrease in the training time for a convolutional neural network model.

We are going to use Convolutional Neural Network (CNN) as well which is a specialized state of multilayer network for detecting the geometric shape in image processing [4]. The role of the CNN is to reduce the images into a form which is easier to process, while maintaining all the features which are critical for getting a good prediction [5]. A CNN extracts features from images instead of manually doing it and this increases the accuracy of our learning models.

If you flatten an image and provide each image pixel as an independent neighborhood, you lose so much important information such as edges or shape of an object. In a block image where we preserve spatial structure, neighbors of pixels maintain that important information. Because of that, we choose to implement our learning algorithm with CNN, which exploits the locality of features to guide the learning of features. In CNN, by cutting the image to small patches rather than processing the full image at a time, we decrease

the complexity and the weights to learn. This is done by convolving our input image and a filter. We used the power of local receptive fields of an image where neighbor pixels are activated and processed together.

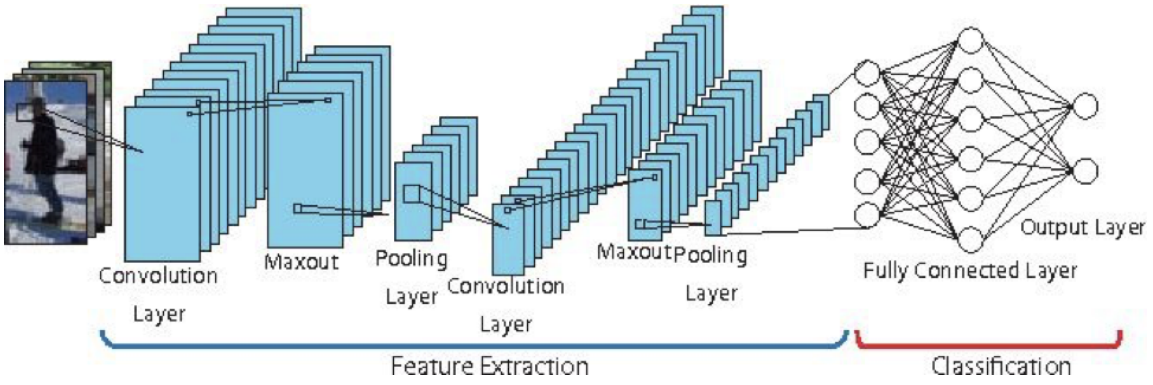


Figure 1. Architecture of Convolutional Neural Network

General architecture of our CNN shows how the action flow is laid out, Convolution layers will be followed by maxout and pooling layers, where we used Max Pooling. Max Pooling chooses the maximum value from the output of feature maps(output of convolution). By choosing the maximum one, we basically summarize the convolution operation in each layer and downsampling our volume.

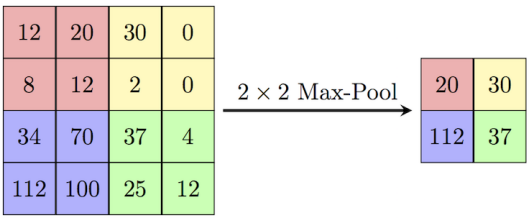


Figure 2. Max Pooling

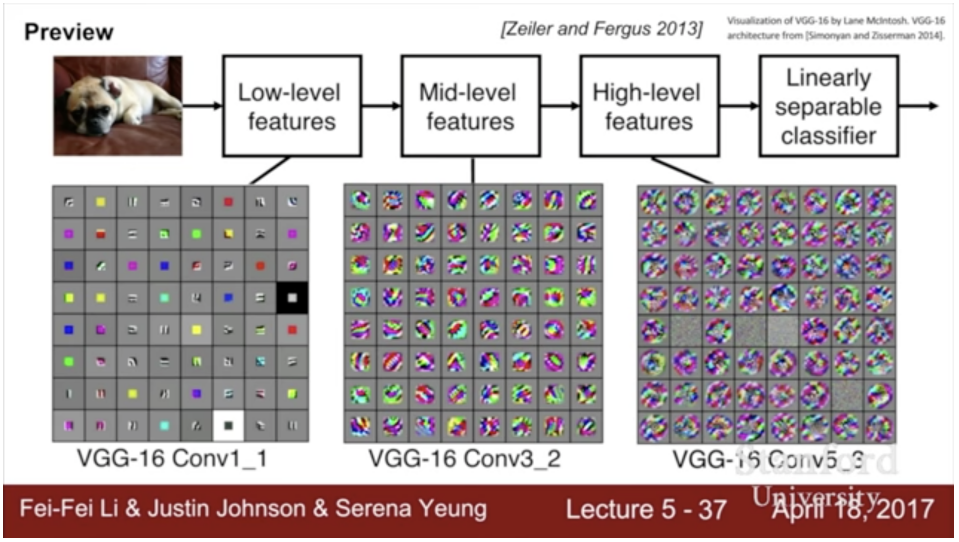


Figure 3. Visualization of Filters

Our CNNs learned the most optimal filters for recognizing specific objects and patterns in our garbage images. In fact, it is learned in each layer by multiple filters. The filters in early layers represent low level features like edges, at the mid level more complex features like corners and at high level things will resemble concepts and blobs. In figure 5, each grid represents a neuron and what the input would look like that maximizes the activation of that neuron.

At the end, after getting our last convolutional layer output, we used a fully connected layer to combine all of these convolutional outputs and use Softmax Function to convert it to a probability distribution. We used the ReLU function (Rectified Linear Unit) as our activation function.

2. Problem Description

Problem: Recycling is a major problem for a zero waste sustainable feature. The process of separating common garbage into different categories of garbage tends to be very repetitive and requires human workforce. Therefore a solution to automate the process of segregating recyclable materials into their respective groups is a major step in the right direction.

Goal: The goal of this project is to make a step towards the end goal of fully automating the process of separating garbage. We aimed to create a model that classifies garbage into different classes.

Method: We used computer vision techniques to handle the problem. We downloaded two datasets from the Internet that contained images of various classes of garbage. We built a model based on ResNet trained on ImageNet. We used transfer learning to utilize this pretrained model.

3. Methods

a. Description of Dataset

Garbage Classification dataset [2] is an image dataset for classifying household garbage. This dataset was chosen since it sorted the garbage into 12 classes while the other dataset [1] consists of approximately 6 classes. The dataset has 15,150 images from 12 different classes of household garbage; paper, cardboard, biological, metal, plastic, green-glass, brown-glass, white-glass, clothes, shoes, batteries, and trash as mentioned while the second one has approximately 2500 images. The datasets have good variety, large quantities, and notes which makes the implementation of the model easier. More data leads to more accurate results, therefore it is crucial to have a large amount of data which is considered while choosing the dataset. Considering this, if the current data is not enough, we are going to combine Garbage Classification dataset with other datasets.

Firstly, we have conducted our experiments with the smaller dataset. Then, we have also used the larger dataset. However, as it can be seen in Figure 2, there is a bias for clothes class. Therefore, we downsampled that class, before using it.

Since each image has a different size, we have resized all images to 256X256, and transformed them into tensors. We then have splitted the data into validation and training sets. The split was performed randomly, with 200 items saved for validation and the rest for training. For now, we have only used the dataset with 6 classes [1] and we calculated the accuracy by using train, validation and test split method.

We have searched for a dataset featuring a large number of images across multiple categories for garbage. We've structured these data into two main folders, namely train and test. The test folder contains approximately 30% of the total images of each category. Additionally each folder contained a number of subfolders equal to the number of classes we have. The reason for making such a structure is to utilize a PyTorch built in function called 'ImageFolder' which is able to read all the data only by providing a path to the parent folder.


BATTERY

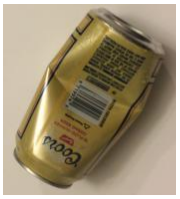
BIOLOGICAL

SHOES


BROWN/GREEN GLASS


CARDBOARD


CLOTHES




METAL

PAPER PLASTIC TRASH

Figure 4: The representation of the Garbage Classification dataset [1]

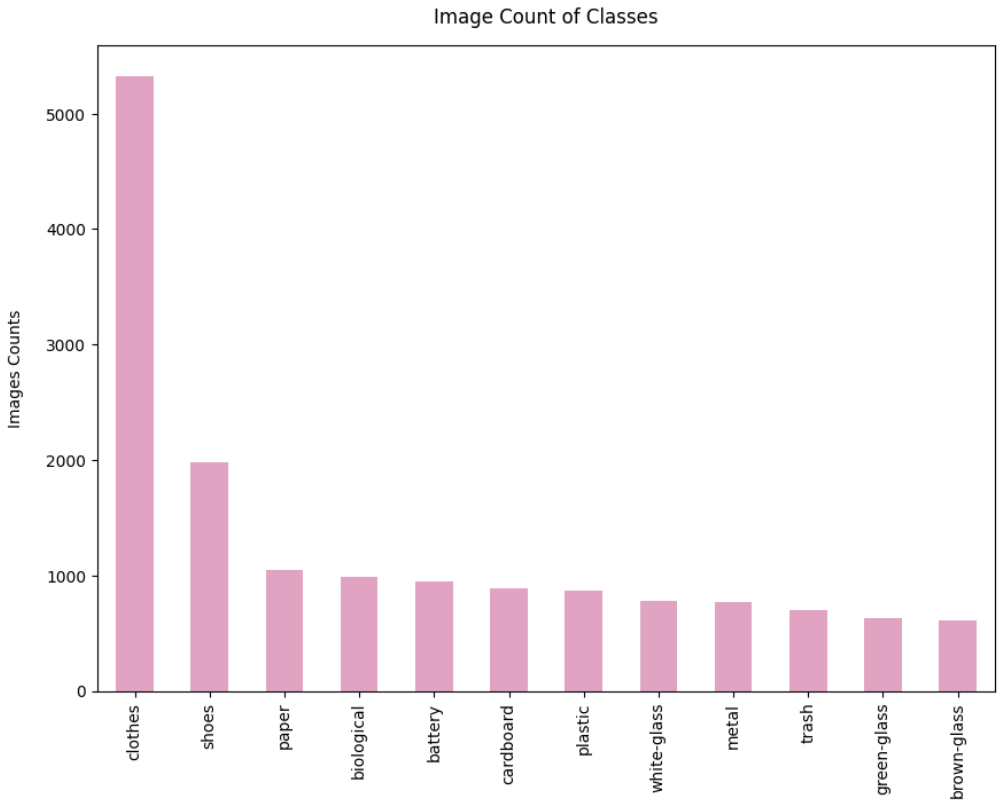


Figure 5: A demonstration of image count of classes for Garbage Classification dataset **before downsampling** clothes class [2]

b. Description of Methods

i. ResNet Trained From Scratch

The first model worked on and experimented was training the ResNet model from scratch, in which all layers are unfrozen and all the layers are trained with our dataset. To unfreeze the layers we had to set preTrained as false. In order to see the difference between training a ResNet model from scratch and training only the last layer, which is the decision layer, 4 experiments were performed. A comparison between ResNet 18 and ResNet50 was observed. The aim of these experiments was to decide the best model and increase the efficiency of our algorithm.

Observation 1: It was noticed that the accuracy of the models when trained from scratch are lower than the pretrained models.

Reason: When training a model from scratch means that we are training all the layers of the model with our dataset, while training a pretrained model, only the decision layer is trained by our dataset and the other layers were already trained by a large dataset. Due to our dataset being not large enough, the accuracy of a model trained from scratch dropped and was seen to be lower than the pretrained model.

Observation 2: It was observed that the ResNet50 model has a higher accuracy and better performance than ResNet18 model.

Reason: Due to the ResNet50 model containing more neural network layers than the ResNet18 model, the ResNet50 trained on more features therefore gets better predictions and higher accuracies.

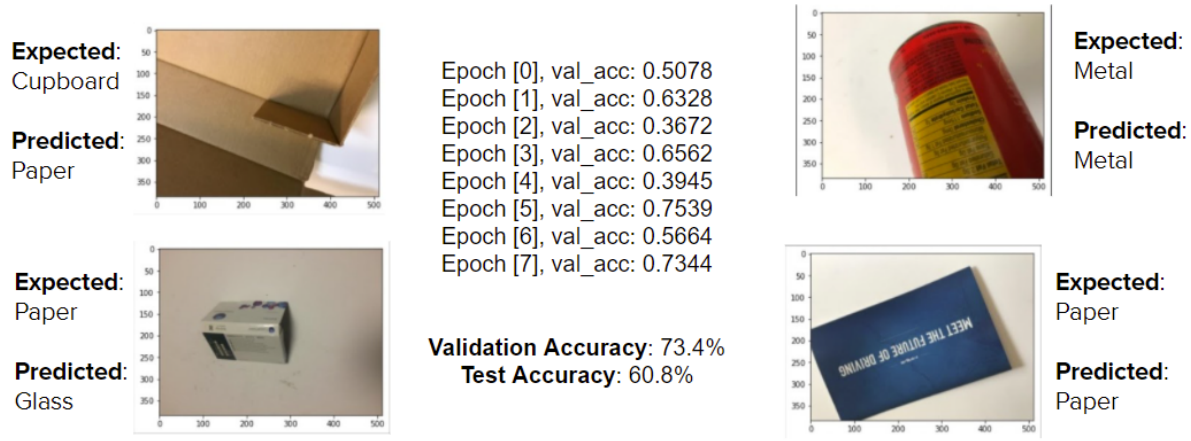


Figure 6: Experiment 1 - ResNet18 and PreTrained = false

In the first experiment a ResNet 18 model was trained from scratch, and the accuracy received from the model was nearly 61%, as shown in figure 6.

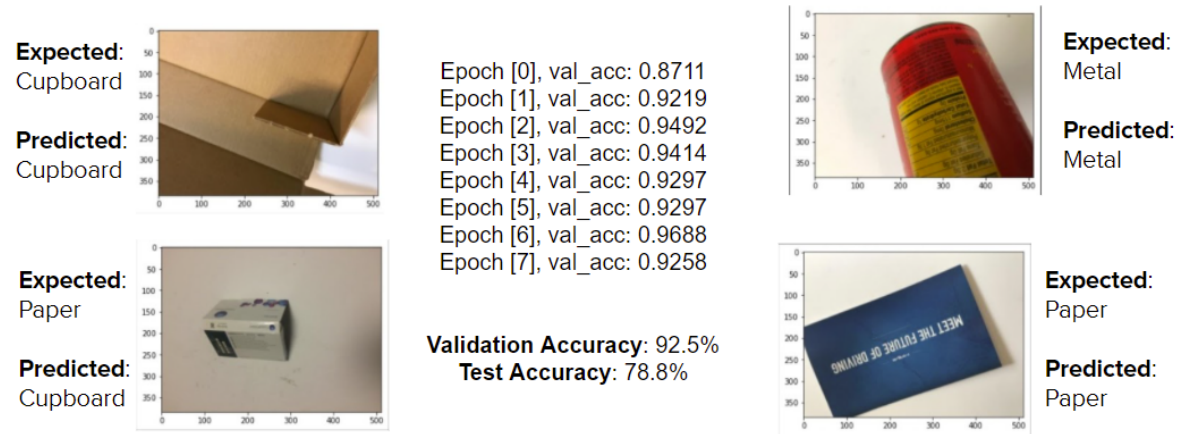


Figure 7: Experiment 2 - ResNet18 and PreTrained = True

In the second experiment, the pretrained ResNet18 model was trained with our dataset, and the accuracy received was approximately 79%, as shown in figure 7.

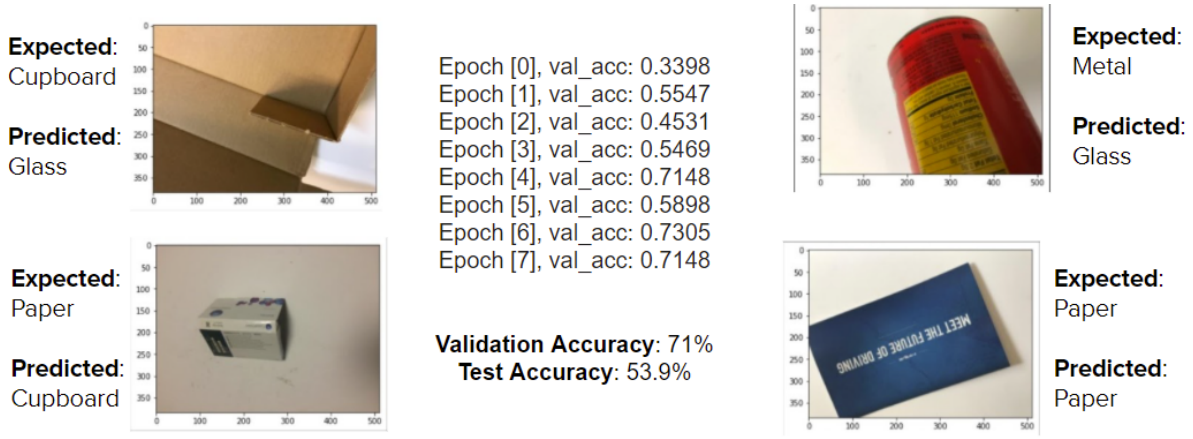


Figure 8: Experiment 3 - ResNet50 and PreTrained = false

The third experiment was training a ResNet50 model from scratch with our dataset, in which the accuracy received was nearly 54%, as shown in figure 8.

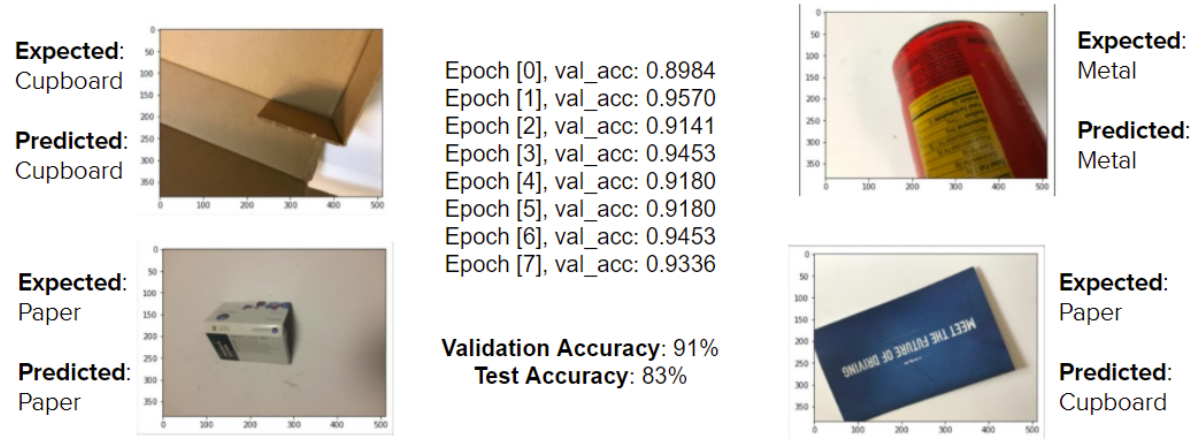


Figure 9: Experiment 4 - ResNet50 and PreTrained = true

The last experiment performed for this method, was training a pretrained ResNet50 model with our dataset, in which the received accuracy was 83%, as shown in figure 9.

Overall, the observation we got from the four experiments performed, were that due to our dataset being too small the accuracy of the pretrained model was always higher than the model trained from scratch, as can be seen when comparing experiment 1 to experiment 2 and experiment 3 to 4. In addition, the ResNet50 model's performance was better than the ResNet18 model, and that is because the ResNet50 model contains more layers than the

ResNet18 model which makes it train on more features therefore getting better predictions and higher accuracies, which can be seen when comparing experiment 2 to experiment 4.

ii. ResNet Without Skip Connections

During the training of a neural network (NN) in the past, the most commonly observed problem was the vanishing gradient problem. The vanishing gradient problem is about not observing any change in the model while training the network. Basically, updating parameters is tried by changing them with smaller Δwi.

$$w_i' = w_i + \Delta w_i \qquad \Delta w_i = -\lambda \frac{\partial L}{\partial \Delta w_i}$$

Figure 10: The calculations for gradient descent

It is possible to understand the vanishing gradient problem by considering the backpropagation algorithm which is about calculating the gradient of the loss function.

$$z = f(x,y) \quad x = g(t) \quad y = h(t) \qquad \frac{\partial z}{\partial t} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial t}$$

Figure 11: The calculations of z based on the neural network parameters

The gradient of a loss function can be expressed by the chain rule (calculation of z based on neural network parameters). The gradient gets smaller, we go backwards.

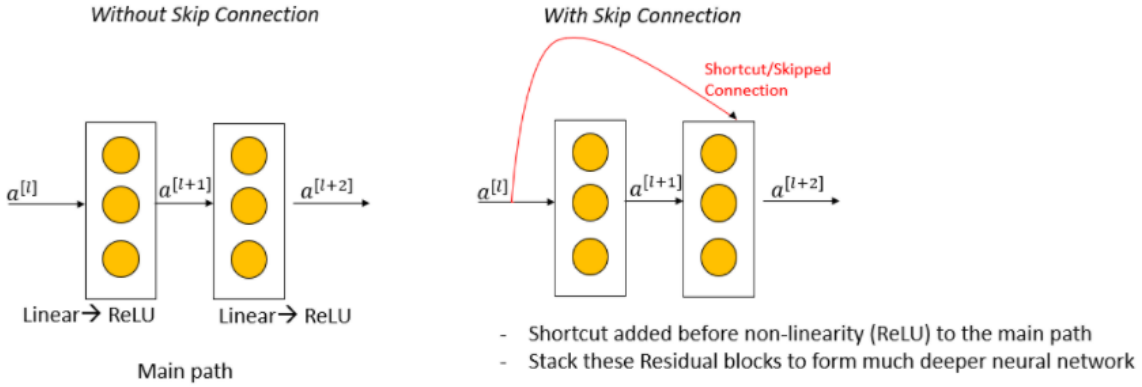


Figure 12: A figure that explains the idea behind skip connections

Skip connections provided an alternative way for the gradient with backpropagation. They skip some of the layers in the NN and direct the output of a layer to the next layers. In Residual Networks (ResNets), by using skip connections, we skip some residual blocks and use the identity function to preserve the gradient's value by multiplying it by 1.

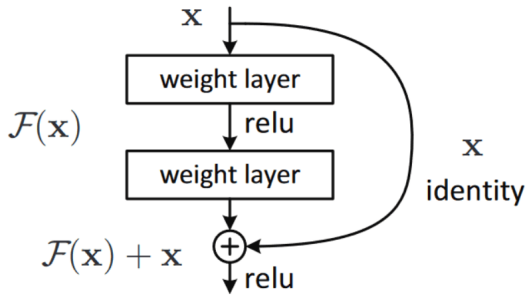


Figure 13: A figure that explains the logic of skip connections and identity function

Without skip connections, the gradient would become too small in the early layers, it might even become zero. In those cases, the early layers would not be updated. Moreover, the features learned in the early layers with lower semantic information would have turned too abstract in the absence of skip connections.

When we made an experiment by removing the residual connections from ResNet50 by deleting the identity layers from forward functions on each block, we observed that accuracy decreased from %60 percent to %43 percent and loss increased from 1.49 to 1.55. Our model is trained on our dataset and transfer learning is not used in this step. Significant changes in these results shows how important residual connections are for ResNet architecture.

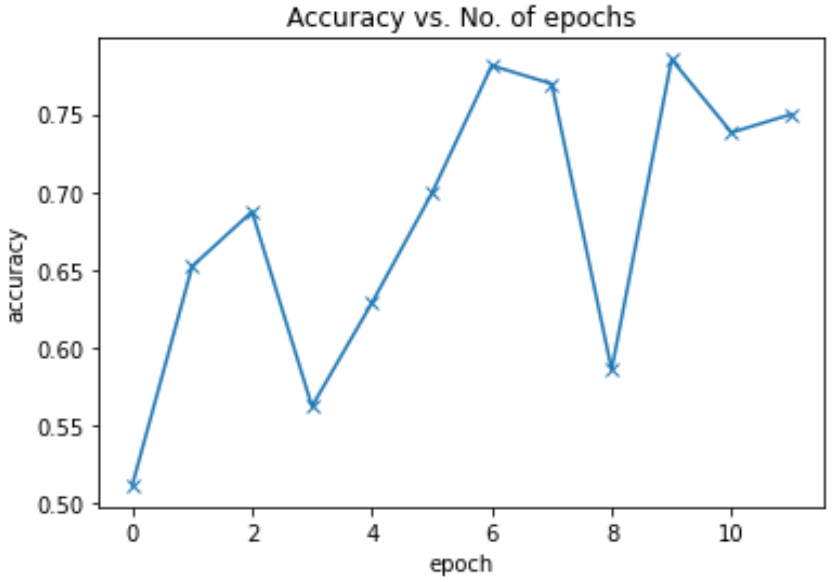


Figure 14. Accuracy per epoch with Skip Connections

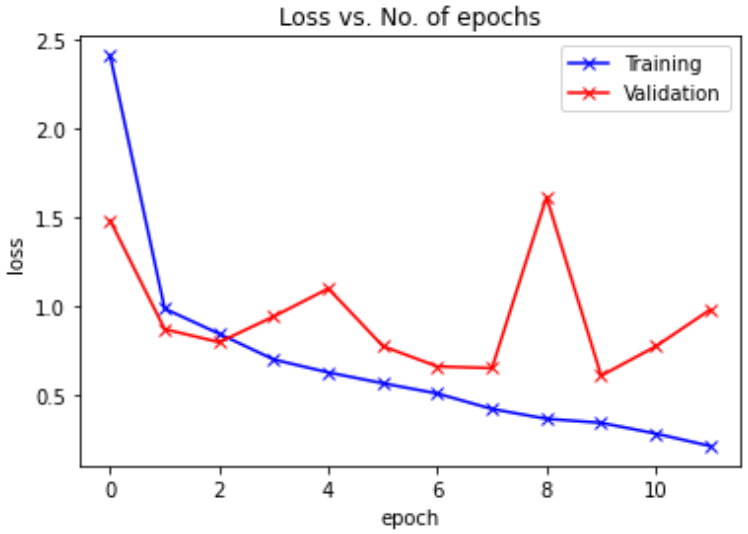


Figure 15. Loss per epoch with Skip Connections

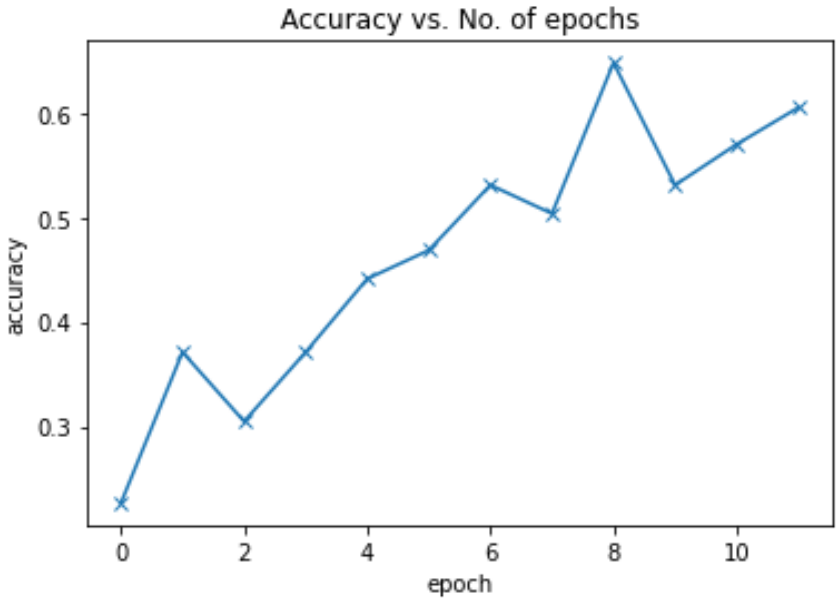


Figure 16. Accuracy per epoch without Skip Connections

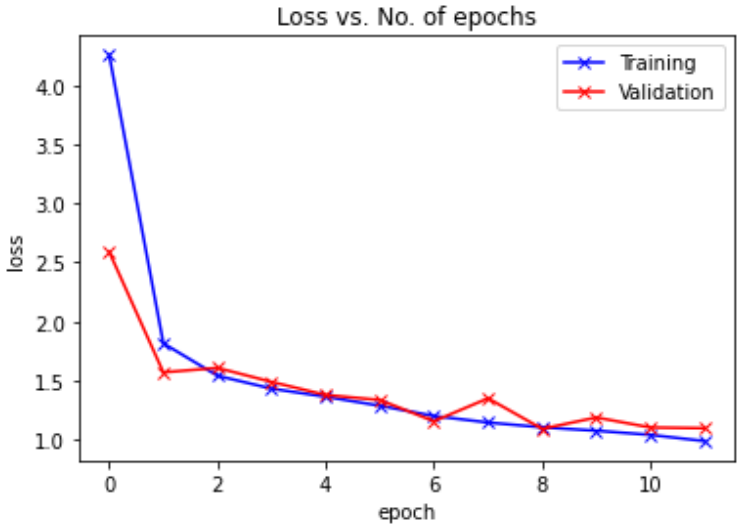


Figure 17. Accuracy and Loss without Residual Connections

iii. ResNet Trained on ImageNet

In order to achieve better accuracy, one of the things we could do was using a pre-trained model on our dataset. We decided to utilize Transfer Learning by using the Resnet50 model trained on the ImageNet dataset, which consists of 1.5 million training images of 1000 different classes. We fine tuned the softmax layer since the pre-trained network on ImageNet comes with a softmax layer with 1000 categories. Our classification task has much less classes.

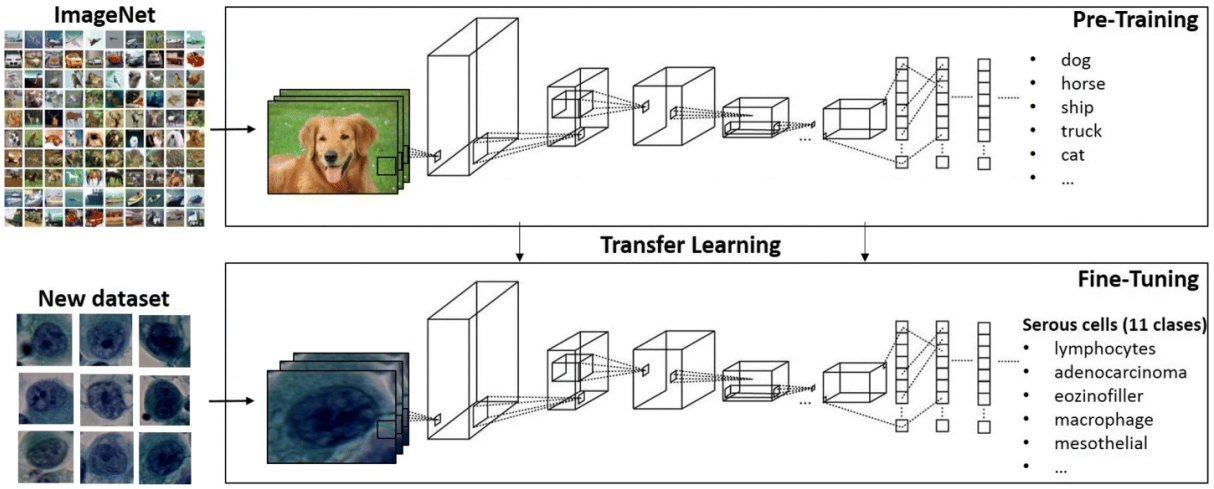


Figure 18. General Idea of Transfer Learning

In order to fine-tune our softmax layer, we replaced the softmax layer on the trained network with a custom softmax layer relevant to our problem. Since we expect the pre-trained weights to be quite good already as compared to randomly initialized weights, in order to not distort them too quickly and too much we used smaller learning rates to train the network. Increased number of epochs(>16) causes overfitting and decrease in

general accuracy with increased loss. Learning rates which are too small do not learn enough by getting stuck, too big learning rates distort the initialized weights and converge too quickly. Optimum result was 8 epochs, learning rate of = 5.5e-5 and Adams optimizer.

Fine-Tuning

NUM_EPOCHS	OPT_FUNC	LEARNING_RATE	Accuracy	Loss
8	ADAM	5.5e-3	%18	1.86
8	ADAM	5.5e-5	%84	1.23
8	ADAM	5.5e-8	%24	1.78
8	SGD	5.5e-1	%18	1.86
8	SGD	5.5e-5	%18	1.98
16	ADAM	5.5e-1	%18	1.88
16	ADAM	5.5e-3	%37	1.72
16	ADAM	5.5e-5	%83	1.21
24	ADAM	5.5e-5	%77	1.27

Figure 19. Experimentation with Fine-Tuning Parameters

In our project, we experimented with Adam optimizer and Stochastic Gradient Descent optimization algorithms to modify the weights of the network in our models. In our experimentation, we noticed that models with Adam optimizer are significantly more accurate than ones with classic SGD. Adam algorithm works better because it maintains a learning rate for each network parameter and distinctly adapts them each epoch.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \qquad \hat{m}_t = \frac{m_t}{1 - \beta_1^t} \qquad \theta_t = \theta_{t-1} - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \hat{m}_t$$

Figure 20. Applying the steps of Adam optimizer

Although the authors of the Adam optimizer recommend against it, the algorithm can be fine tuned by modifying these four parameters:

- ➔ α. Alpha is the initial learning rate of the algorithm. This parameter is also known as the step size. Weights are updated by this proportion. Larger values tend to yield faster learning. Default value recommended by the inventors of the algorithm is 0.001.
- ➔ β_1. The exponential decay rate for the first moment estimates. Usually initialized at 0.9.
- ➔ β_2. The exponential decay rate for the second-moment estimates. Usually initialized at 0.999. The algorithm performs better with computer vision problems when the beta2 parameter is set closer to 1.
- ➔ ε. This parameter is designed to rationalize division by zero cases in the implementation. It is set to 1e-8 by default.

4. Results

To get a better understanding of the accuracy of our model, we have plotted the loss values per an epoch for each of the test and validation data. We would expect the curves for both of the test and validation losses to be decreasing and close to each other. If the training curve decreased initially, and started increasing while the validation curve kept decreasing, that is an indication that some overfitting has happened. Based on the Figure 21, we notice that both of the curves are decreasing and close to each other; hence, overfitting is minimal in our model.

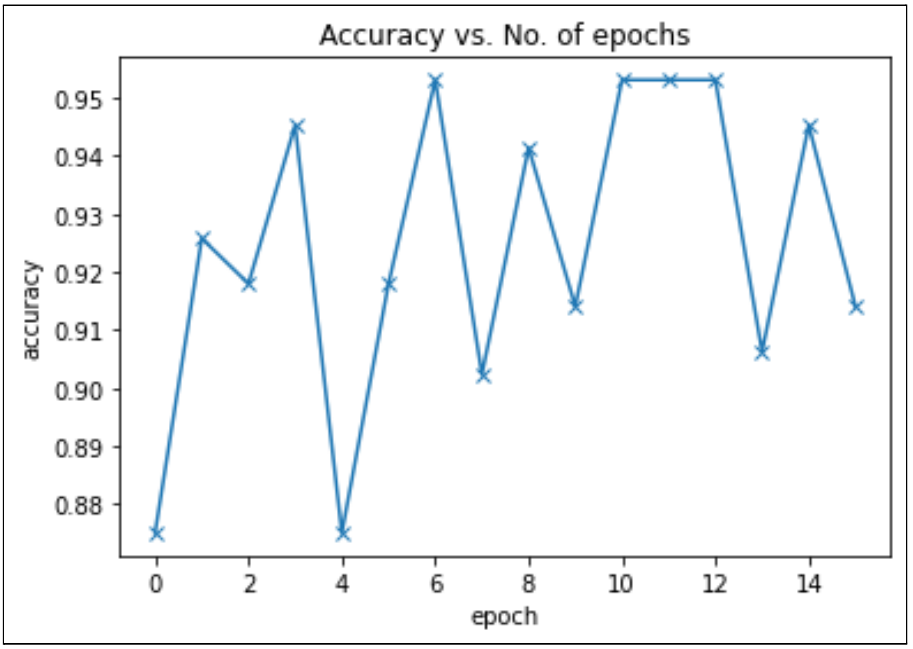


Figure 21: Accuracy per epoch

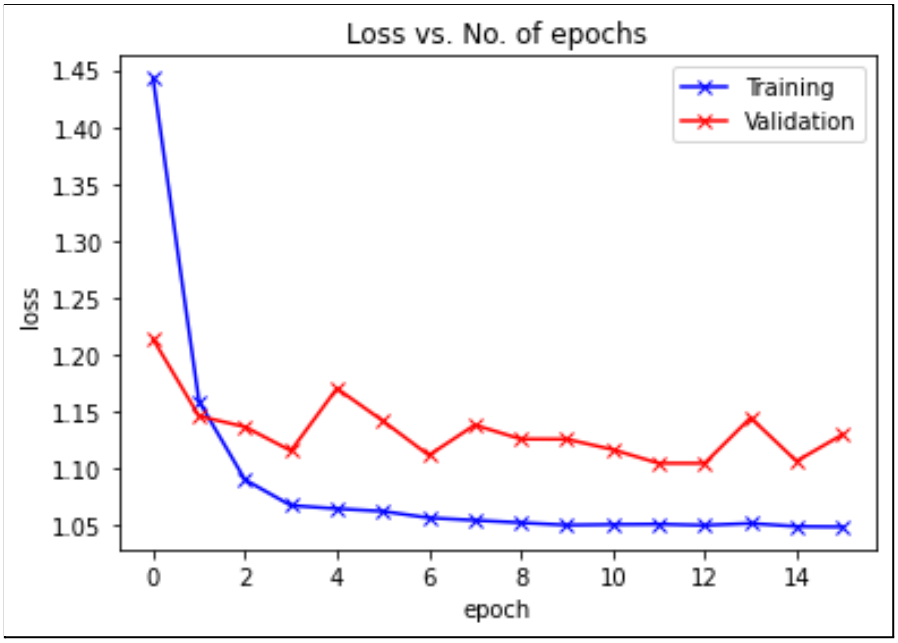


Figure 22: Loss value per epoch using Training and validation data

By testing the model on training data we have achieved an accuracy of 84.93%, which is actually pretty good given our limited dataset. In order to increase the accuracy, we are planning to use data augmentation.

Data Augmentation

Data augmentation is a powerful technique for utilizing smaller datasets. This technique generates randomly transformed images from the original images from the dataset [6]. Therefore we can generate some more variety in the existing dataset. By utilizing this technique we want to achieve a better accuracy score with our modest-sized dataset. The below results are obtained after using data augmentation.

```
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.p
  cpuset_checked))
Epoch [0], train_loss: 2.0021, val_loss: 1.8412, val_acc: 0.7188
Epoch [1], train_loss: 1.8423, val_loss: 1.8510, val_acc: 0.7070
Epoch [2], train_loss: 1.8067, val_loss: 1.8228, val_acc: 0.7344
Epoch [3], train_loss: 1.7897, val_loss: 1.8051, val_acc: 0.7734
Epoch [4], train_loss: 1.7815, val_loss: 1.8080, val_acc: 0.7617
Epoch [5], train_loss: 1.7713, val_loss: 1.8302, val_acc: 0.7422
Epoch [6], train_loss: 1.7627, val_loss: 1.8352, val_acc: 0.7344
Epoch [7], train_loss: 1.7548, val_loss: 1.7757, val_acc: 0.8398
```

Figure 23: Train and validation loss and validation accuracy per epoch

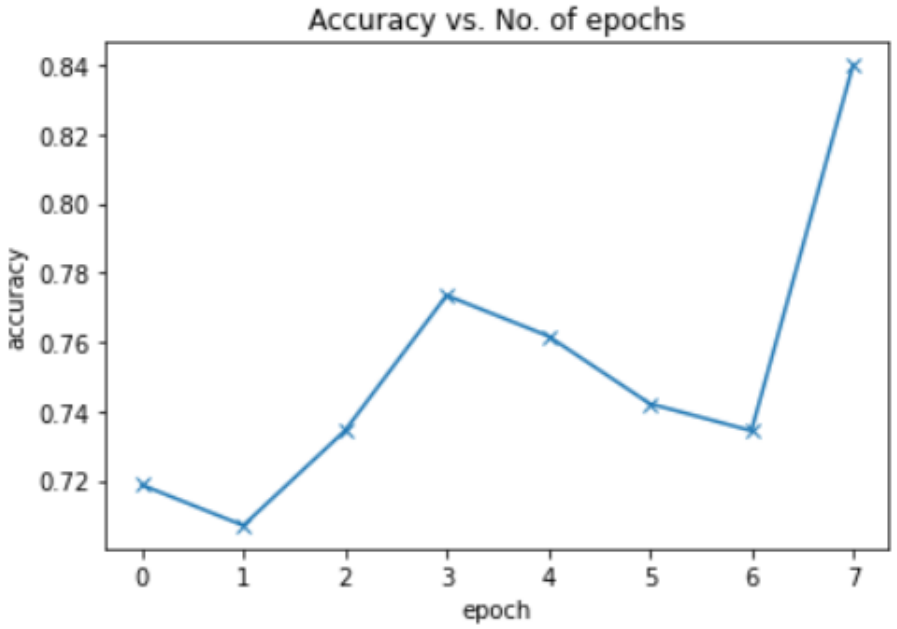


Figure 24: Accuracy per epoch

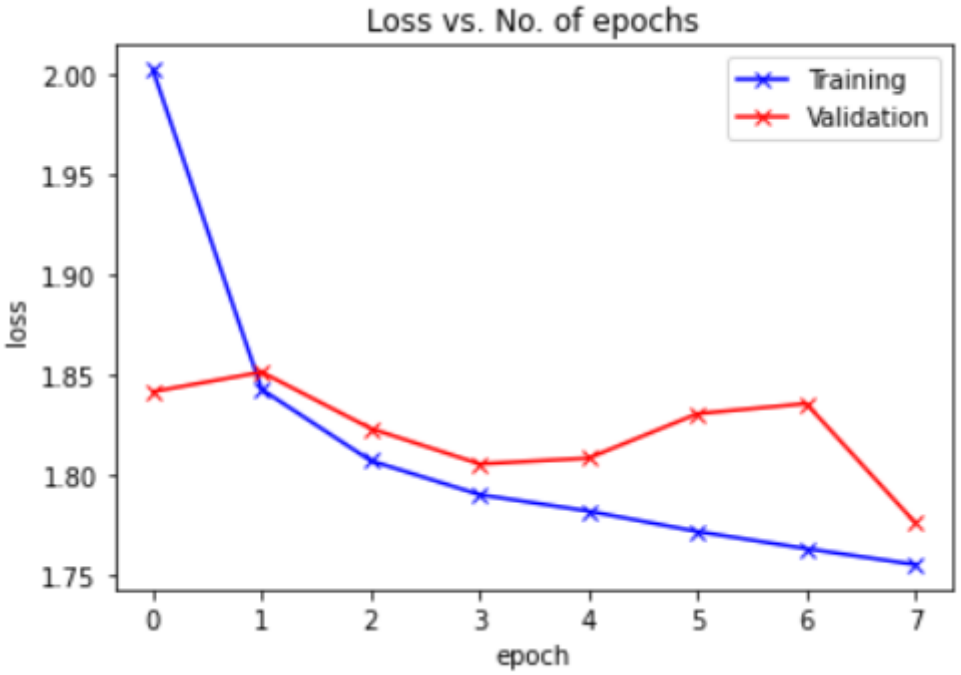


Figure 25: Loss value per epoch using Training and validation data

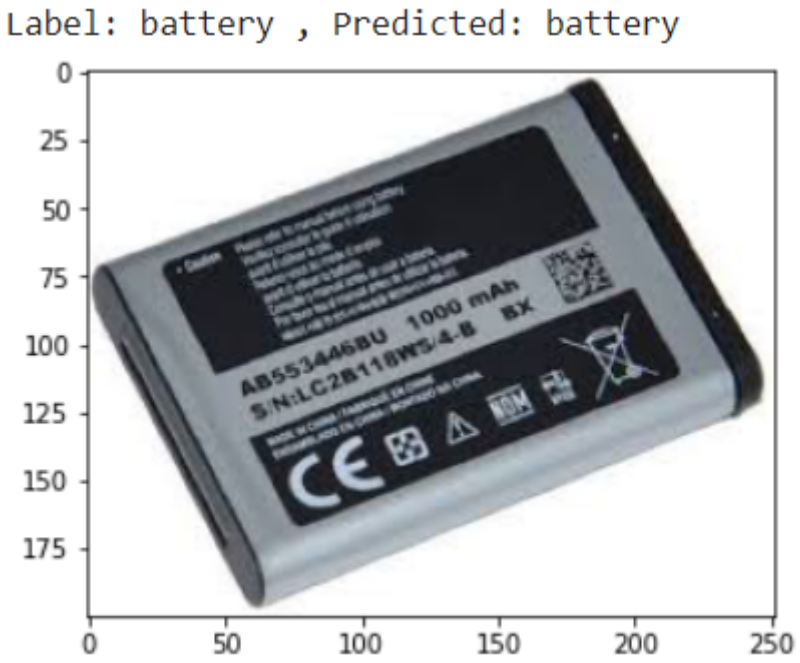


Figure 26: An example from the experiment conducted with larger dataset using data augmentation

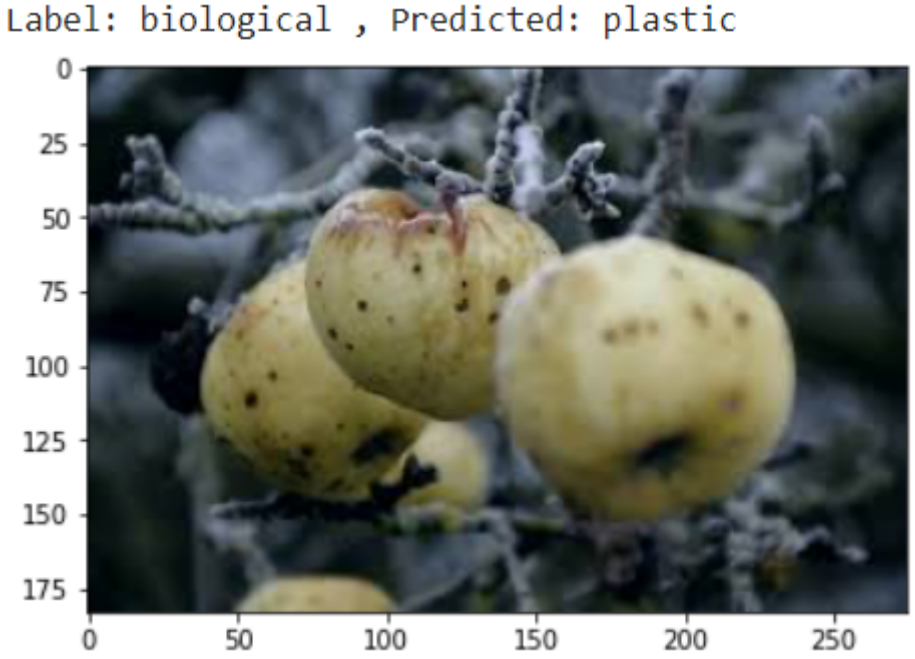


Figure 27: An example from the experiment conducted with larger dataset using data augmentation

```
{'val_acc': 0.603398323059082, 'val_loss': 1.9751793146133423}
```

Figure 28: The validation accuracy and the validation loss with larger dataset using data augmentation (Model 1)

FEATURES					
ResNet Layer #	preTrained	Skip Connection	Val. Acc.	Test Acc.	Loss
ResNet18	TRUE	TRUE	92.50%	78.80%	1.3
ResNet18	FALSE	TRUE	73.40%	60.80%	1.44
ResNet50	TRUE	TRUE	91%	83%	1.23
ResNet50	FALSE	TRUE	71%	53.90%	1.47
ResNet18	FALSE	FALSE	72%	55%	1.79
ResNet50	FALSE	FALSE	60%	43%	1.55

Figure 29. Experiment Results with ResNet

From Figure 29 we can observe that without Skip Connections ResNet performance decreases significantly with increased loss and decreased validation & test accuracy. Pre Trained models perform much better than custom models since they work on better datasets.

5. Discussions

It was observed that the ResNet architecture performed well on image classification tasks. The reason ResNet performs better than most of the other CNN architectures is that it provides a solution to the Vanishing Gradient problem that prevents the weight from modifying its value and stops the neural network from further training. Also Adam(Adaptive Moment Estimation) Optimization Algorithm performs better compared to Stochastic Gradient Descent.

Adam Optimizer	**Stochastic Gradient Descent**
Three hyperparameters (β_1=0.9, β_2=0.999, $\varepsilon=10^{-8}$), adaptive learning rate α	Single, static learning rate (α)
Faster training performance	Fast compared to Batch GD but still slow
Higher accuracy	Low accuracy

6. Conclusion

To conclude, we can infer Transfer Learning helps to increase the accuracy of the model since the model uses knowledge from another model which is trained with probably larger

datasets which requires enormous computation power, time and energy sources. ResNet performs very well on image classification tasks and skip connections increase the accuracy by eliminating vanishing gradient problem and ensuring that the performance of the higher layer is at least as good as the lower layers.

After observing the power of Transfer learning, in the future we can work on other learning methods and their areas of use to create more effective and efficient algorithms. In the future, the garbage classification model can be improved in a way that it detects the categories of multiple objects at the same time. In order to achieve this, multi-task learning can be used. Thus, we might focus on those methods in the future.

7. Appendix

Team Members	Contributions
Işık Özsoy	Investigation of possible issues of Garbage Classification dataset Researched on skip connections and how to remove them from the ResNets implementation
Utku Kalkanlı	Researched on transfer learning and how CNN is used on classifying images. Researched on CNN architectures and ResNet. Conducted experiments with the Resnet50 model trained on the ImageNet dataset and fine tuned the softmax layer. Worked on implementation of ResNet without residual(skip) connections.
Yahya Mohamed	CNN research to understand how it will be used, implementing some models using the CNN approach to classify the garbage Conducted experiments with a pre trained and not pretrained ResNet50 models
Mohammed Yaseen	Researched on state of the art garbage classification solutions Conducted experiments with a pre trained and not pretrained ResNet18 models Implemented data augmentation and confusion matrix
Oğuz Tüzgen	Investigated new datasets to increase the scope of the model Python library research Researched on optimization algorithms which are Adam optimizer and Stochastic Gradient Descent optimization algorithms and finding one with the best accuracy

Table 1: Contribution of each team member

8. References

[1] “Garbage classification,” Kaggle, 24-Nov-2018. [Online]. Available: https://www.kaggle.com/asdasdasasdas/garbage-classification. [Accessed: 27-Oct-2021].

[2] M. Mohamed, “Garbage classification (12 classes),” Kaggle, 24-Jan-2021. [Online]. Available: https://www.kaggle.com/mostafaabla/garbage-classification. [Accessed: 27-Oct-2021].

[3] Sangminwoo, “RecycleNet/Data/dataset-resized at master · sangminwoo/recyclenet,” *GitHub*. [Online]. Available: https://github.com/sangminwoo/RecycleNet/tree/master/data/dataset-resized. [Accessed: 27-Nov-2021].

[4] U. Ozkaya and L. Seyfi, “Fine-tuning model comparisons on Garbage Classification for Recyclability,” *NASA/ADS*. [Online]. Available: https://ui.adsabs.harvard.edu/abs/2019arXiv190804393O/abstract. [Accessed: 26-Nov-2021].

[5] S. Saha, “A comprehensive guide to Convolutional Neural Networks - the eli5 way,” *Medium*, 17-Dec-2018. [Online]. Available: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Accessed: 26-Nov-2021].

[6] V. Lyashenko, “Data Augmentation in python: Everything you need to know,” *Data Augmentation in Python: Everything You Need to Know*, 12-Nov-2021. [Online]. Available: https://neptune.ai/blog/data-augmentation-in-python. [Accessed: 27-Nov-2021].