Derek Sadler
12/8/2024
CPTS 437 Machine Learning

Project Write Up

For my project the objective is to construct a dataset that features various household items and then utilizes a conventional neural network model (CNN) for accurate identification and categorization of each item. I then utilize this model for predicting images when placed in places with different backgrounds. The reason I chose this approach is that I wanted experience with making my own item identification system and wanted to get an understanding on how adaptable my model could be when making predictions when images are placed in different positions or backgrounds.

When working on this project I aimed to split the work into various sections starting with research, then I moved onto collecting my data, and then making my project. The reason I approached it this way is that it allowed me to slowly ease my way into understanding how to build a complicated CNN model.

Research:

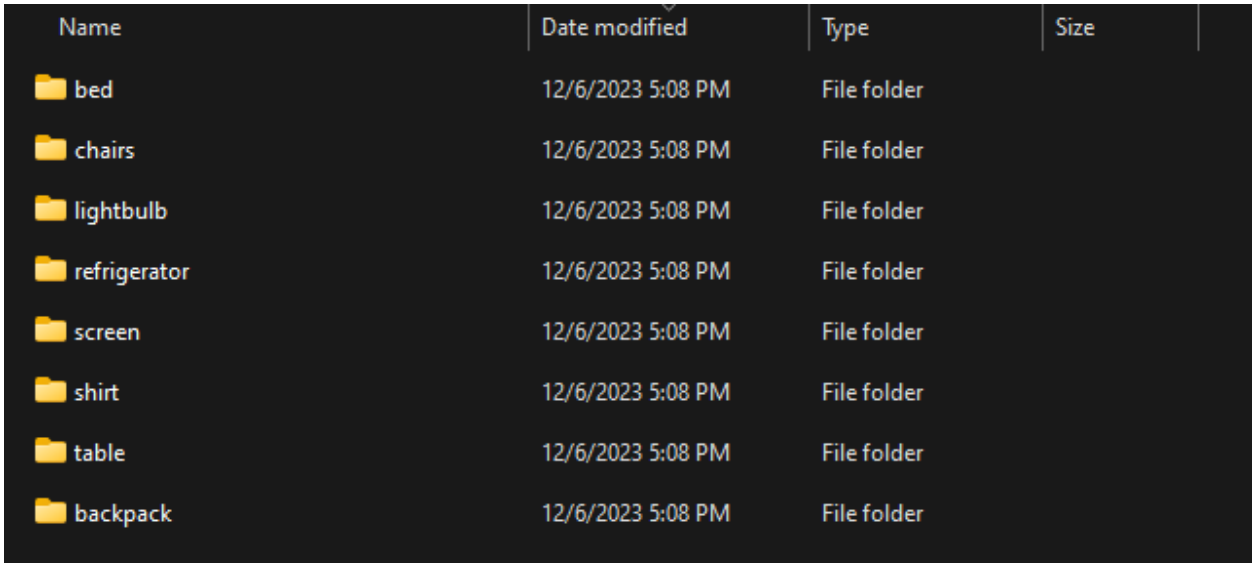
Initially starting this project I utilized a few datasets from robotflow.com in order to learn what kind of data is required for making a custom dataset. I found mostly every data set comes in a format where there are many unique images for training with fewer for testing. The reason for this is that we need to be sure our model is able to make accurate predictions when receiving new data so providing it with lots of training images can help the model understand what patterns it needs to look for, then for the testing images can be of the same object but from different positions in order for the model to be able to be tested if it can make accurate predictions on an objects identification.

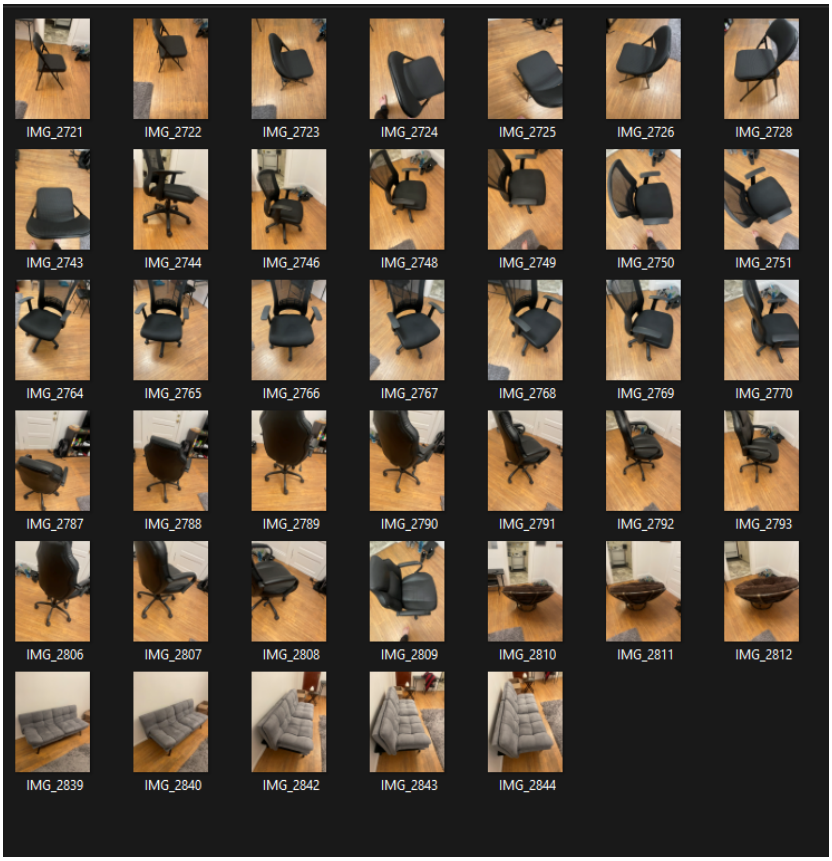
After learning about the process of making a data set I gave myself a background on how to make a Convolutional Layered model (CNN). I primarily referred to sources that could give me a background or basic code examples on how I could construct a model. The process of making my model will be discussed later and my sources can be found at the bottom of this document. From analyzing these datasets and model tutorials I then was able to move onto making my own dataset and model.

Data Collection:

For my data collection I had the goal to get various objects throughout my house and take pictures of them for the training, testing, and validation of my CNN model. I collected a total of 1600 images for my dataset split between 8 different objects. Each object had a total of 200 images collected for them with 100 of them being used for training, 25 for validation, and an additional 75 for testing. For testing I had three

different types of backgrounds that the object was placed on. This left me with 25 images for each background in my training data, the backgrounds being no background, blue, and red. The objects that I chose for my data were my backpack, bed, chairs, lightbulbs, refrigerator, electronic screens, shirts, and tables.





When taking the pictures of my objects I made sure to take as many photos of them from different angles to ensure there are variations in the images. I also tried to take a variation of the same objects that I could find, such as different types of chairs, lightbulbs for their respective dataset type. I took those photos from my phone and then uploaded them to my google drive to store for use. I then took those photos and converted them to a json format to save data and prepare them for reading in the CNN. The overall structure of that data was in folders based on their use in modeling, such as training, validation, or testing types.

A limitation with my approach for data collection is that my dataset only contains items from my home. This limitation left me only a few of the same items for my dataset, for instance I only have one bed, one refrigerator ect, and a handful of chairs/tables. This left me with concerns that my overall CNN predictive accuracy would be affected due to not having variation in the data. Another limitation I came across for this project was that even though I had a lot of images for every object it still may not be an effective amount for accurate data prediction when it comes to machine learning. Machine learning requires a large amount of data for training and even though I took 200 images for each object it still may not be enough to make a model that can identify objects based on different backgrounds.

These issues could all be fixed by adding more images to my dataset but that would require me to take more pictures of the same object or collect images from online sources, however, having more images will slow down my training process and could not be in line with the data I had in mind for my project. To remedy those issues for my training model I took experiments where I trained my model with all the objects, and half my objects in order to see if my model would get more accurate predictions.

Data Preprocessing:

After collecting my data and organizing I then was able to start the coding of my project. To start I imported all the libraries I required for my project, this included basic libraries such OS for operating system commands, numpy for number processing, matplot and seaborn for the GUI, tensor flow for processing the data and modeling and sklearn to go with it. After setting my libraries I then started setting my data generators for my training, and validation data set. The data generator is a utility that is used to help generate batches of image data and is able to augment my existing data to be more varied. The augmentation rescales my image and randomly modifies the image for training through factors such as changing the image rotation, width, height, and zoom. Then I set the flow path of the data generator as well as the target size of the images being read, and the batch size that determines the number of images in each batch that will be fed to the model. For the class mode of my generators I set each to a categorial

mode which means that my generator will always return a one hot encoded label which is used for the classification task of my model.

Model Generation:

For generating the model I first set it to a sequential one from karas. The model is a layeled one where layers are added sequentially one after the other which the layers can read from the previous layers. Then I set multiple convolutional layers to the model starting with one that consists of 32 filters with a size of 3x3 done through a ReLU activation function. The ReLU function essentially returns any positive inputs from the model. For each convolutional layer a pooling function is used to downsample the data by taking the maximum value of an input window, this also helps in reducing the computation complexity of the model.

The reason each convolutional model is added is because each layer increases the models ability to learn more complicated features about the inputted image which helps it identify patterns in the images. For instance in the first layer I have only 32 filters which can help my model examine basic features in the image, while in deeper layers I have 64 and 128 features which can identify increasingly more complex features in the model. The filters act as the hyperparameters for my model and I decided the values since I didn't want to make my model more complex and the general practice for CNN models appears to have this doubling process for its filters.

After setting my layers I then flatten the model to set my 3D vector to a 1D one so that its data is set for the fully connected layers. I then set my fully connected layer with 512 nodes for the CNN. After that I set my final layer to have 8 neurons for each of my classes and utilize a softmax activation function, the function allows for my raw output of my model to be transformed into vectors of probabilities which can be used for my image classification. Finally I can compile my model to the training process, I set my optimizer to adam which will be used to determine how my model will update its parameters during training, I set my loss function to categorical cross entropy for training for multi classification, and then I set set my metrics to accuracy for evaluating the training. With this set up I can train, evaluate and make predictions for my model.

Model Training:

For training my model I set it to fit the training generator with various other settings set to make sure my model is trained as I expected. The model trains on 100 images for each object, there was a total of 20 epochs for training. Additionally I set a validation generator for the training in order to prevent possible overfitting on the model, there were a total of 25 validation steps for each image. On average every epoch takes a total of 44 seconds for testing and validation with the total increasing when more

images are added. Once my model is trained I then collect the accuracies for my training data and validation data (refer to data results to see graph).

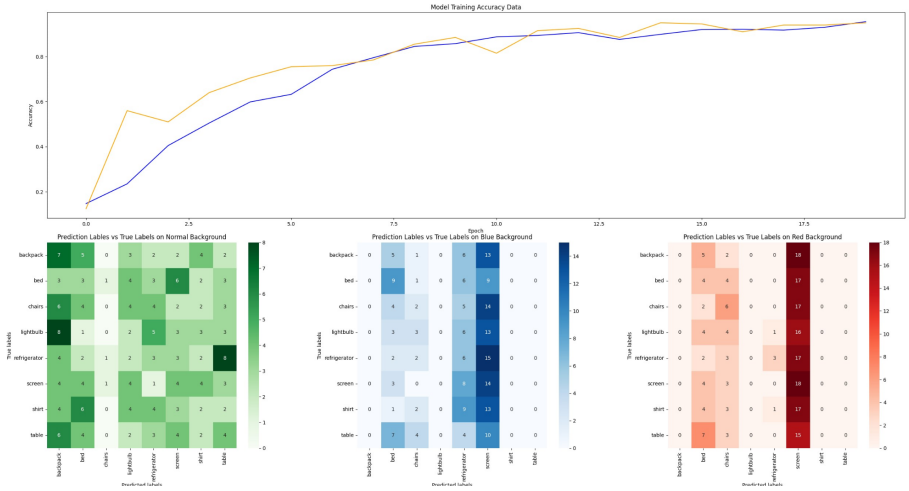
Data Results:

In order to represent my models accuracy I created a heatmap of my true images vs predicted images. Since each object has 25 testing images for every background type the matrix is ideally expected to have 25 images predicted for each image with the true image being the same as the predicted.

For my data results most of the predictions in my confucian matrix were not accurate to my true results. The predictions had some cases where they were more accurate to the expectation but for most of the images the prediction vs actual image types were scattered throughout the matrix. This was especially the case when the background for the testing were different which resulted in usually one image being the predicted outcome. However, despite my matrix accuracy being questionable my overall evaluation of my images were accurate unlike my predictions with only the normal background.

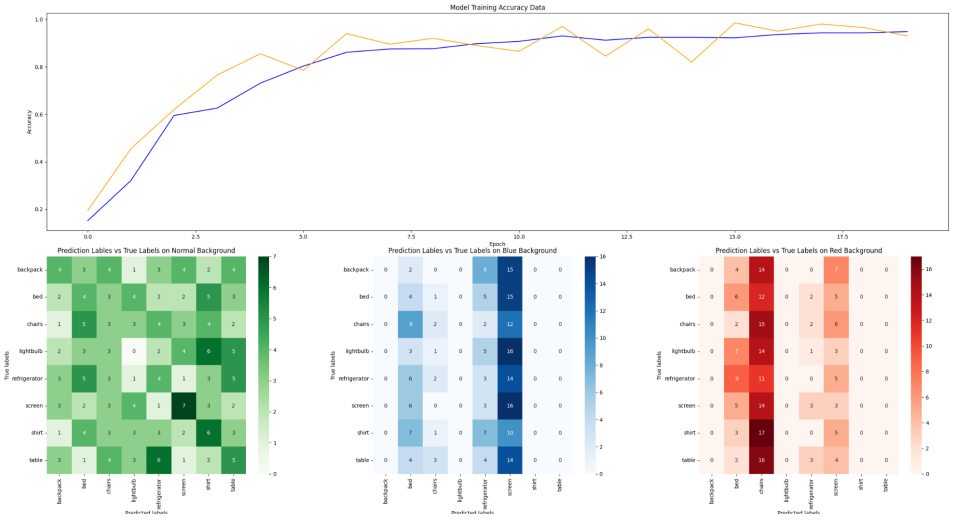
8 Items Evaluation Scores

```
Normal Background
Evaluation Accuracy: 90.00%
Evaluation Loss: 23.07%

8/8 [==============================] - 8s 967ms/step - loss: 22.3273 - accuracy: 0.2650
Blue Background
Evaluation Accuracy: 26.50%
Evaluation Loss: 2232.73%

8/8 [==============================] - 8s 1s/step - loss: 35.3133 - accuracy: 0.3550
Red Background
Evaluation Accuracy: 35.50%
Evaluation Loss: 3531.33%
```



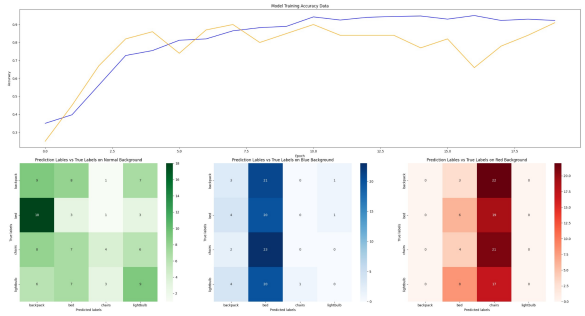
The reason my results could be like this could be due to a data mismatch or my model potentially overfitting. My model is accurately able to evaluate my data but is not accurate in predicting new data which could mean my model has trouble identifying unseen patterns in data or variation in it which could mean it's not adaptable enough. For this project I have tested a few solutions and made a few alterations to my training process to see if that could have an impact on the accuracy of my predictions. To start with I first tested how adding my testing data to my training data would impact my models accuracy in predicting. Due to this not being the best practice I used it only for testing to see if my model would be accurate when being fed already existing data. My results were slightly improved but it did not seem to verify my model was able to make accurate predictions.

Testing Data Set in All Items

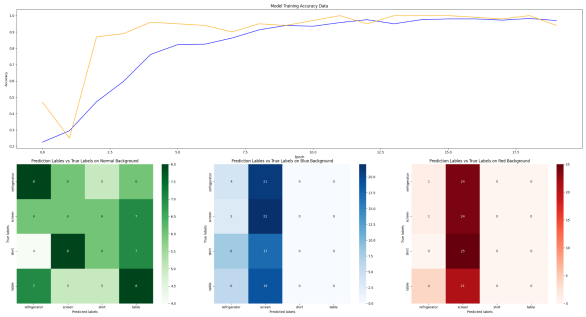


Another approach I made was to test my model on a lower amount of objects to see how my model would perform when it has to make predictions on fewer objects so that the ones represented are more likely to be accurate. Unfortunately I had the same result with my true predictions vs actual predictions being inaccurate roughly half the time for each dataset variation I used.

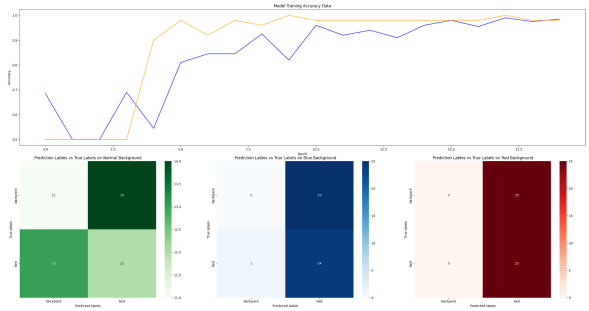
Backpack, Bed, Chairs, Light Bulb Only:



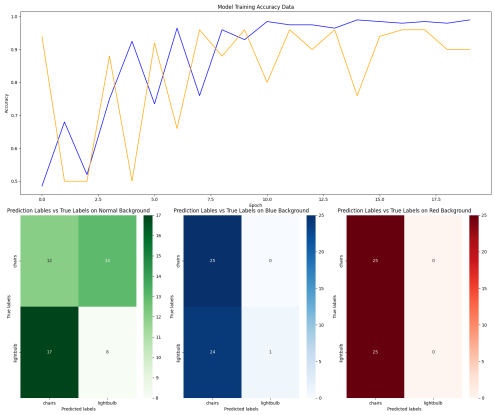
Refrigerator, Screen, Shirt, and Table Only:



Backpack vs Bed



Chairs vs Lightbulb



Screen vs Refrigerator

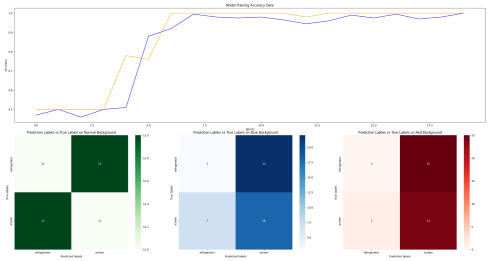
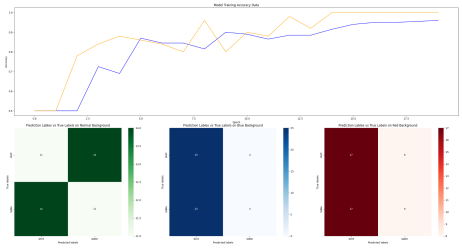
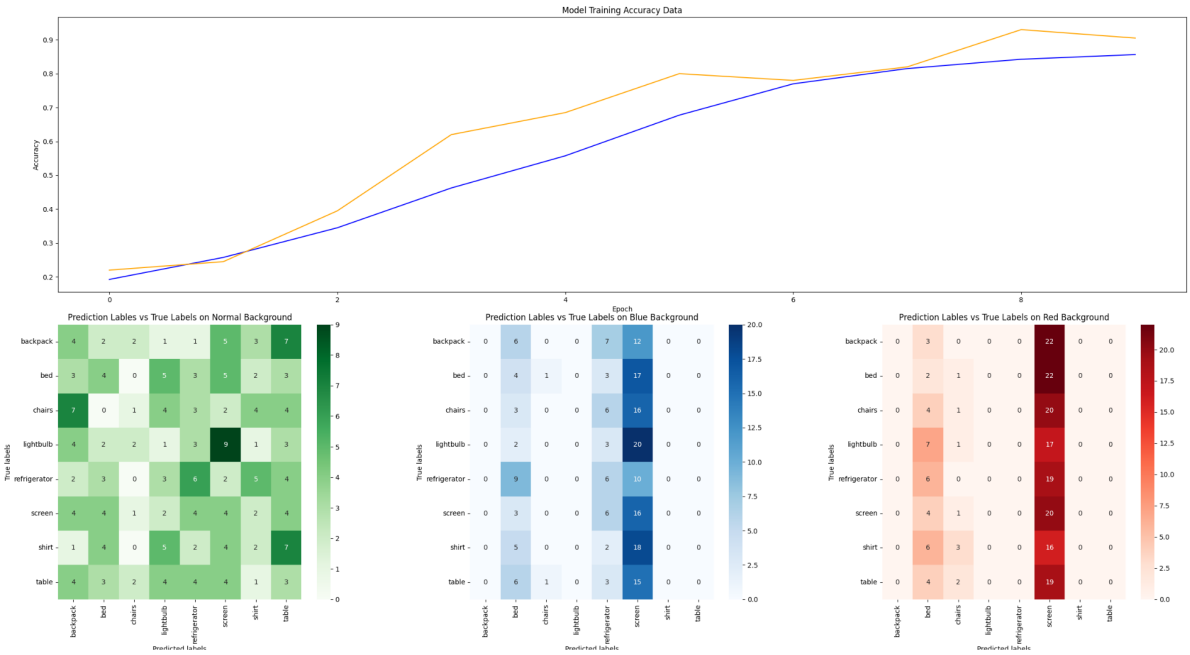


Table vs Shirt



My third approach was to simplify the model so that any possible situation of overfitting would not occur in the model. This could allow for my model to possibly be able to adapt to new pieces of information better. After testing the simplified model I noticed no noticeable change in my models accuracy. With it only being roughly 12.5% accurate unlike my normal model that is roughly 20% accurate.




```
Normal Background
Evaluation Accuracy: 82.50%
Evaluation Loss: 52.38%

8/8 [==============================] - 8s 970ms/step - loss: 15.1834 - accuracy: 0.2250
Blue Background
Evaluation Accuracy: 22.50%
Evaluation Loss: 1518.34%

8/8 [==============================] - 8s 1s/step - loss: 11.1846 - accuracy: 0.2000
Red Background
Evaluation Accuracy: 20.00%
Evaluation Loss: 1118.46%

8/8 [==============================] - 8s 991ms/step
Normal Background Accuracy: 12.50%
```

Conclusion:

For this project I found that even though my model at times was able to gradually improve with its ability to evaluate a training data set and accurately identify a training dataset the results typically showed that the model became too overfitted in nature meaning it is not able to recognize objects when the image is different. This is most noticeable when an image that is in a different background is fed into the model for prediction. For future work in this project I have multiple methods that I could use to fix the models ability to accurately predict the objects. One solution is to get more training data for each object, currently I only have 100 training images for each object and that results in my dataset being fairly small due to other datasets having thousands of images to train on. Despite the trouble I came across for some aspects of the image

identification I learned how to create a CNN model and develop the features required to make a model that can predict an image based on training data. I'm confident If I had a more varied dataset with more images for training the majority of the issues I came across for my results could be fixed.

Project References:

“Give Your Software the Power to See Objects in Images and Video.” Roboflow, roboflow.com/. Accessed 7 Dec. 2023.

Rosebrock, Adrian. “Keras Conv2d and Convolutional Layers.” PyImageSearch, 8 June 2023, pyimagesearch.com/2018/12/31/keras-conv2d-and-convolutional-layers/.

Sharma, Pulkit. “A Comprehensive Tutorial to Learn Convolutional Neural Networks from Scratch (Deeplearning.Ai Course #4).” Analytics Vidhya, 14 July 2023, www.analyticsvidhya.com/blog/2018/12/guide-convolutional-neural-network-cnn/.