Multi Playing Card Recognition Final Report

Brian Wei
bwei1@andrew.cmu.edu

Jerry Yu
jerryy@andrew.cmu.edu

1 Introduction

Our project is Multi Playing Card Recognition. Our system will recognize any number of playing cards and then classify each of their suit (Spades, Clubs, Diamonds, and Hearts) and rank (2-10, Jack, Queen, King, Ace). The input to our system will be an image with 1-5 (n) of cards. The output will be an n long list of suits and ranks. We will evaluate our overall system with f1 score, the harmonic mean of precision and recall. With f1, we can evaluate our accuracy, our false positive rate, and true negative rate in one metric.

The data we will use will be generated ourselves programatically. We have built a script to generate any number of cards on any background with random size, position, and rotation. The only invariant we have during data generation is that one corner of a playing card must be visible.

We have created two datasets to evaluate our system on. Dataset 1 is generated from combinations of 52 standard playing cards on different backgrounds. The second dataset is generated from 3 decks with unique card styles with randomly adjusted brightness and contrast. The second dataset is made to more closely replicate real life conditions.

Recognizing cards allows fully autonomous agents to play card games. In the real world, agents will not be given cards on the table, but will have to recognize them with their own vision systems. In addition, we were interested in exploring how to build a vision system invariant to rotation, heavy occlusion, brightness, and various backgrounds.

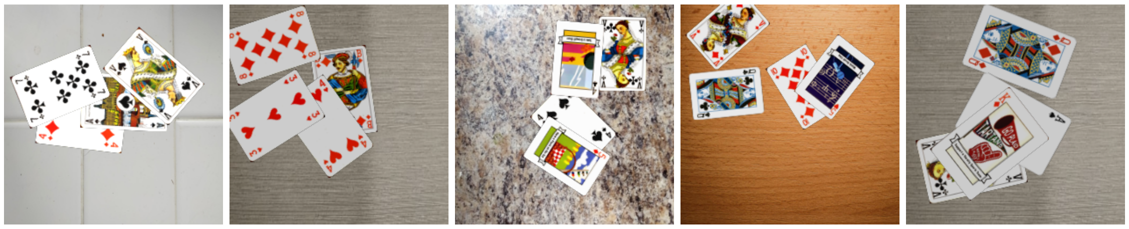


Figure 1: Images from dataset 2 (realistic dataset)

2 Background

In the midway report, we created a script to generate the clean dataset 1. For our baseline methods, we used two models. The first is a CNN regression model that counts the number of cards. For each number, we applied a CNN classification model to classify each card. For both CNN models, we used ReLU, max pooling, dropout, a fully connected layer at the end, and a softmax for the classification models. The models were trained on 13,000 images and the model with the best validation accuracy was chosen. Our overall baseline model had a low 0.168 f1 score.

It was surprising to see that the total f1 score was less than the product of the f1 scores of the two methods (0.8 and 0.5). We believe that the baseline had poor performance because the classification step depended on the counting step to be successful. If the counting step fails, the classification models would return a random output.

3 Related Work

Several previous strategies for card classification rely on adding bounding boxes to cards present in an image first, before starting classifying the cards. Pimentel and Bernadino [1] first identify where cards are using edge distributions, then rotate the card to the upright position. They use two methods of classification. One is using a probabilistic rigid method looking at edge features and choosing the class that maximizes the MLE of the features. The other method is having independent parts of the cards analyzed and using a voting scheme from a distribution of each part. The paper reports an accuracy of 1.0 under ideal conditions, with a recognition rate of 0.6 with 20 percent brightness variation.

Additional strategies include using multiclass logistic regression (MLR), multiclass support vector machines (MSVM), and convolutional neural networks [2] for classification, but the team only in recognized one card at a time. They report around 0.887 accuracy for MLR and 0.844 accuracy for MSVM. They report 1.0 accuracy for using CNNs, but that is limited to one card, not multiple, like our task.

Another technique used for image classification are residual networks which allows for the construction of substantially deeper convolutional networks, while preventing vanishing and exploding gradients through shortcut connections [3]. We did not find literature that applied ResNet on playing card datasets.

We wanted to perform image segmentation to mask out the background before classifying the cards. We chose to follow an autoencoder based method used in SegNet. It uses a convolutional encoder and decoder to segment the image into multiple regions. [4] [5] It's performance varies across multiple datasets, but its best performance has around 0.9 f1 score.

4 Methods

4.1 Baseline approach: Counting and Classification

We started off by evaluating our model on the clean dataset 1. The f1 scores will be from there. We discuss the baseline method in the Background section of the report.

4.2 Multilabel with binary cross entropy

We worked with our mentor Abhishek to combine both of our models into the multilabel model. The multilabel model takes in an image and outputs a binary 52 dimension vector. Each dimension represents a card, a 1 means the card is present, a 0 means it is not.

The model has 3 convolutional layers with relu activation, 2x2 max pooling, and 0.25 dropout. At the end, the convolutional output is flattened and passed through a fully connected layer to a 52 dimension vector. The vector has a sigmoid function to use the outputs as probabilities. If the card probabilities are above 0.5, they are present (1).

To prevent the overfitting with our baseline, we trained with more data, using 50,000 images and evaluating on 5,000 images. We used adam optimizer with 0.001 learning rate and binary cross entropy as the loss function since the output can have any number of ones and zeros. Multilabel model had a f1 score of 0.338, which was better than our baseline.

4.3 Multilabel with weighted binary cross entropy

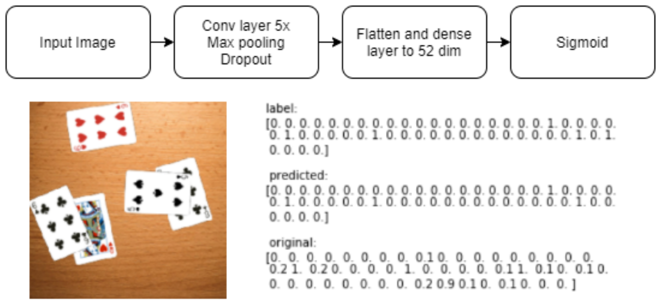
We realized that the labels were very sparse, with at most 5 ones and the other 47 being zeros. So, the model would return all zeros as the output and still have a low loss. The low loss would result in a lower gradient and the model would not learn.

We wanted to give more weight to the model predicting a card was present rather than predicting a card is not present. To do this, we multiplied the loss from ones in the label by 5 and multiplied the loss from ones in the prediction by 3. With this change, our model reached a f1 score of 0.391.

4.4 Multilabel with more layers

Our models were still overfitting, with a train f1 score of around 0.74 and a test f1 of 0.391. This is from large number of parameters in the model (2,855,220), with most of the parameters were from the final fully connected layer from the flattened output to the 52 dim output (2,798,900). With more conv layers, the image size is reduced from max pooling, and the final dense layer was much smaller. Each conv layer also has few parameters from parameter sharing so using 5 conv layers instead of 3 conv layers reduces the total number of parameters to 213,428.

With 5 layers, our model reached a validation f1 score of 0.902, with the training f1 score close at 0.855. The model was still able high enough bias while reducing variance. We also tried reducing the number of conv layers to 2, which had more parameters and 0.2620 lower f1 score. This verifies that more conv layers leads to less parameters and better performance.



```
label:
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.
 0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 1.
 0. 0. 0. 0.]

predicted:
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.
 0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.
 0. 0. 0. 0.]

original:
[0.  0.  0.  0.  0.  0.  0.  0.  0.1 0.  0.  0.  0.  0.  0.  0.  0.  0.
 0.2 1.  0.2 0.  0.  0.  0.  1.  0.  0.  0.  0.  0.1 1.  0.1 0.  0.1 0.
 0.  0.  0.  0.  0.  0.  0.  0.  0.2 0.9 0.1 0.  0.1 0.  0.  0. ]
```

Figure 2: Multilabel 5 layers architecture with 0.914 f1.
Included an input, label, prediction, and original probabilities

4.5 ResNet

On top of the multilabel with weighted binary crossentropy, we wished to see if we attain even better results by using a larger, deeper network. However, by simply continuing to add more convolutional layers (with pooling and dropout), we run into the vanishing gradient problem as we cannot effectively propagate gradients through the entire network. Thus, instead of straight convolutional layers, we use a residual network [3]. This allows us to learn using a model with larger capacity.

We wanted to use residual networks with 50 and 152 layers respectively, but this led to a substantial challenge when training these models. Because of the large size of these models, they require more data and more computational resources to train. As such, we were only able to train for 15 epochs per model per dataset.

4.6 Realistic data for dataset 2

We were satisfied with the multilabel model with dataset 1, but we wanted the model to learn to recognize cards in more realistic images. So, we created dataset 2 by using 3 decks with unique card faces. We also introduced randomized brightness (0.7 - 1.1) and contrast (± 20) changes on the images. This was done by the linear transformation of random brightness * image + contrast. Our 5 layer multilabel model had around 0.829 f1 score with dataset 2, we confirmed that dataset 1 was more difficult than dataset 1 and there was room for improvement in our model. The f1 score mentioned in the following methods will be from the realistic dataset 2.

4.7 Image segmentation autoencoder preprocessing

The idea of image segmentation preprocessing is to develop a model that extracts the cards from the background of the image. We wanted to use a specialized model to remove the noise of the background so that our multilabel classification model is able to focus on classifying the cards accurately. We followed the model of SegNet [4] [5], using a autoencoder with 5 conv layers with max pooling for encoding and then 5 conv layers with upsampling for decoding. We used relu activation for each layer and a final conv layer to generate the mask.

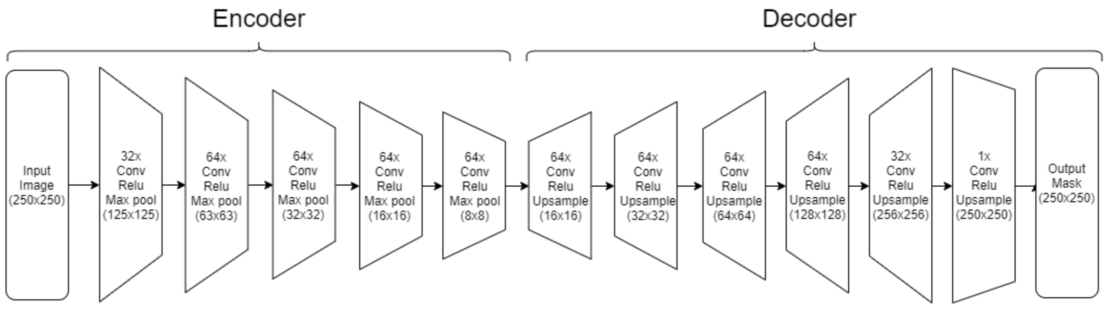

Encoder
Decoder
Input Image (250x250)
32x Conv Relu Max pool (125x125)
64x Conv Relu Max pool (63x63)
64x Conv Relu Max pool (32x32)
64x Conv Relu Max pool (16x16)
64x Conv Relu Max pool (8x8)
64x Conv Relu Upsample (16x16)
64x Conv Relu Upsample (32x32)
64x Conv Relu Upsample (64x64)
64x Conv Relu Upsample (128x128)
32x Conv Relu Upsample (256x256)
1x Conv Relu Upsample (250x250)
Output Mask (250x250)

Figure 3: Segmentation autoencoder architecture
Each layer has output dimensions labeled

The input to the segmentation model is the image and the output is a binary mask of the same size which represents if the pixel is in the background (0) or foreground (1). Like the multilabel model, we evaluated image segmentation using f1 score of the flattened mask, which factors in both our model recall and precision. We originally used cross entropy loss like Segnet [4] [5], but we had similar sparsity issues in our output like with the multilabel model (all zeros). So, we used weighted cross entropy to weight the errors in the foreground more than the errors in the background. This was also to conservatively include all the foreground, even while leaving in some background.

Overall, the performance of the segmentation autoencoder model was good, with 0.976 f1 score for both datasets of 50,000 images over 20 epochs. The risk of applying a preprocessing segmentation model is that it may cover our important information we need for classification, but manual review showed that the segmentation still left all the rank and suit information.

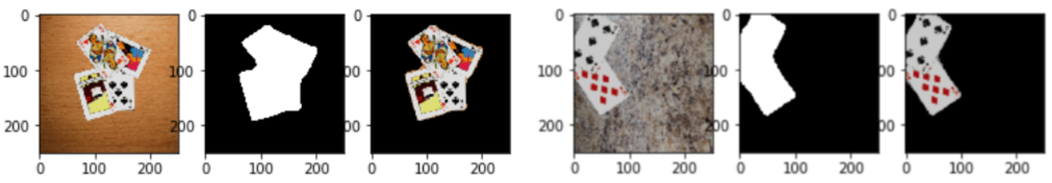
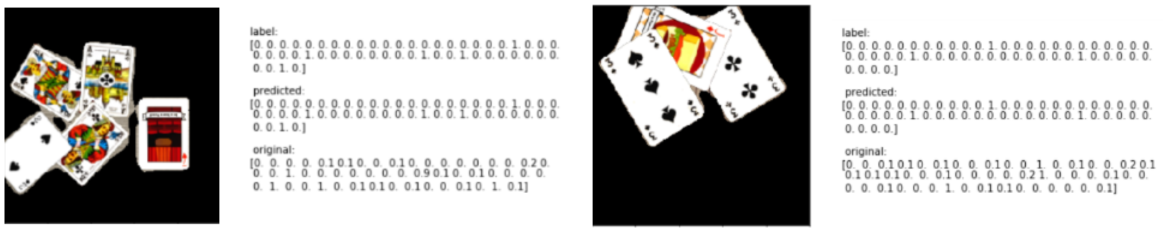


Figure 4: Input image, Output mask, Mask applied to input

After training the segmentation model, we preprocessed all the data by applying the foreground mask. We then trained the best 5 layer multilabel model on the preprocessed data. We saw an improvement on dataset 2, with a f1 score of 0.912.



```
label:
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.
 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 1. 0.]

predicted:
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.
 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 1. 0.]

original:
[0.  0.  0.  0.  0.1 0.1 0.  0.  0.1 0.  0.  0.  0.  0.  0.  0.  0.2 0.
 0.  0.  1.  0.  0.  0.  0.  0.  0.  0.  0.9 0.1 0.  0.1 0.  0.  0.  0.
 0.  1.  0.  0.  1.  0.  0.1 0.1 0.  0.1 0.  0.  0.1 0.  1.  0.1]
```

```
label:
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.
 0. 0. 0. 0.]

predicted:
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.
 0. 0. 0. 0.]

original:
[0.  0.  0.1 0.1 0.  0.1 0.  0.  0.1 0.  0.  1.  0.  0.1 0.  0.  0.2 0.1
 0.1 0.1 0.1 0.  0.  0.1 0.  0.  0.  0.  0.2 1.  0.  0.  0.  0.1 0.  0.
 0.  0.  0.1 0.  0.  0.  1.  0.  0.1 0.1 0.  0.  0.  0.  0.  0.1]
```

Figure 5: Example classification with segmentation
Included segmented input, label, prediction, and original probabilities

5 Results

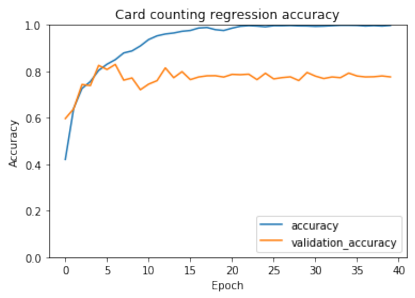
Overall Model Validation F1 Score		
Model	Dataset 1 (clean)	Dataset 2 (realistic)
Multilabel with 5 layers and weighted cross entropy (80 epochs)	0.902	0.829
ResNet 50 layers (15 epochs)	0.904	0.892
Segmentation and 5 layer multilabel model (80 epochs)	0.765	0.912

Overall Model Validation F1 Score (40 epochs)		
Model	Dataset 1 (clean)	Dataset 2 (realistic)
Counting and classification (Baseline)	0.168	
Multilabel with 3 layers and binary cross entropy	0.338	
Multilabel with 2 layers and weighted cross entropy	0.262	
Multilabel with 3 layers and weighted cross entropy	0.391	
Multilabel with 5 layers and weighted cross entropy	0.853	0.761
Segmentation and best multilabel model	0.738	0.879

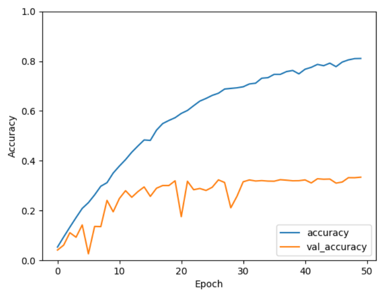
Image Segmentation Validation F1 Score		
Model	Dataset 1 (clean)	Dataset 2 (realistic)
Autoencoder background segmentation	0.976	0.976

5.1 Baseline Approach: Counting and Classification

Card counting accuracy



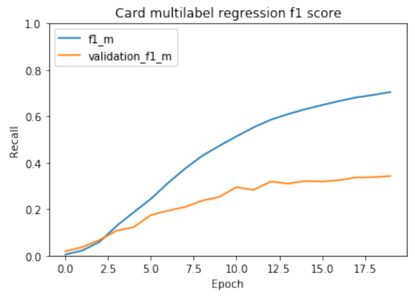
Four card accuracy



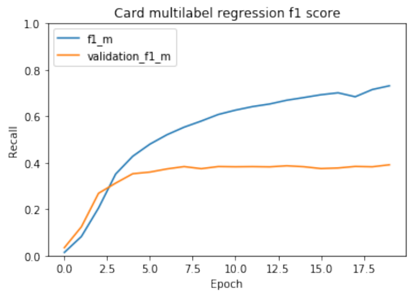
We discuss the baseline results in the Background section. We got .168 f1 score with overfitting, shown by a large difference between training and data.

5.2 Multilabel Binary Cross Entropy vs Weighted Cross Entropy

Binary cross entropy f1 for dataset 1



Weighted cross entropy f1 for dataset 1



We decided to use f1 score as the metric to evaluate the multilabel model. This is because the labels are very sparse and mostly zero. So, the model that predicts all zeros (no cards) still had relatively high accuracy. F1 score balances true negatives and false positives, so it helped give an overall perspective on the performance of our system.

Both of the multilabel models were trained on a dataset of 50,000 cards and evaluated on 1,000 images. We found that increasing the size of the dataset helped improve overfitting. Both used the adam optimizer with 0.001 learning rate.

The weighting of the cross entropy increases the loss for getting a one wrong (a card prediction) rather than a zero wrong (a card is not present prediction). This helped increase our validation f1 score from 0.338 with binary cross entropy to 0.391. However, we still saw a significant difference between training and validation performance, which showed that our model was capable of understanding the task but was overfitting. This model was around our expectations because it incorporated both of our previous models with almost double the performance.

5.3 Multilabel 5 Layers

We used the same adam optimizer, the same 50,000 dataset, and evaluated on the same metric of f1 score. However, increasing the number of layers from 3 to 5 decreased the number of parameters by 5x due to parameter sharing and the reduction of the dimensions of convolutional output. With the results, we see that the f1 score and the loss from both the training and validation set are very close. This shows that overfitting is

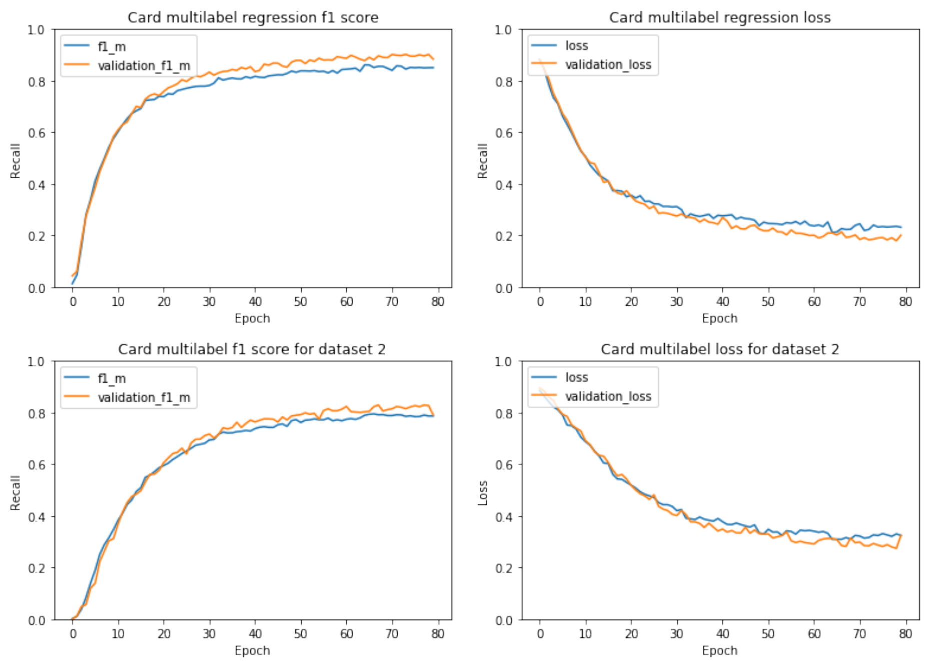


Figure 6: Multilabel 5 layer results: 1st row from dataset 1, 2nd row from dataset 2

very low once we reduce the number of parameters. This is due to reducing the overall variance of the model while still having enough parameters to capture the bias of the data.

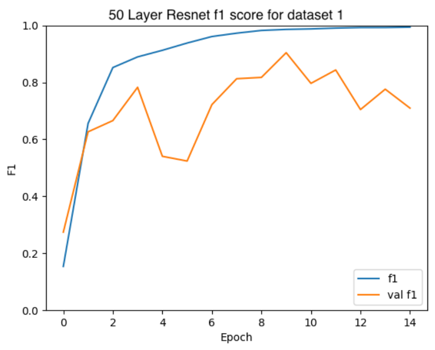
We reached 0.902 f1 score on dataset 1 and 0.829 f1 score on dataset 2. This model exceeded our expectations, it was surprising that changing the number of layers to reduce the number of parameters increased our performance by 2x. Overall, reducing the number of parameters helped us find the tradeoff between bias and variance. The model performed worse on the more realistic dataset 2 than on dataset 1, but still had very similar training and loss curves. This was in line with our expectations as dataset 2 had much more noise with multiple decks and random brightness.

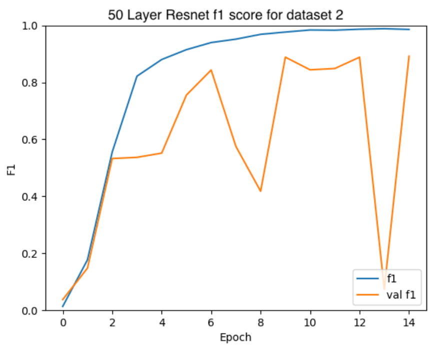
5.4 ResNet

We again use the same 50,000 dataset, optimizer, and evaluation metrics, for the residual network. By using a ResNet with 50 layers, we reached 0.904 f1 score on dataset 1 and 0.892 f1 score on dataset 2. These models were only trained for fifteen epochs each due to temporal and computational limitations.

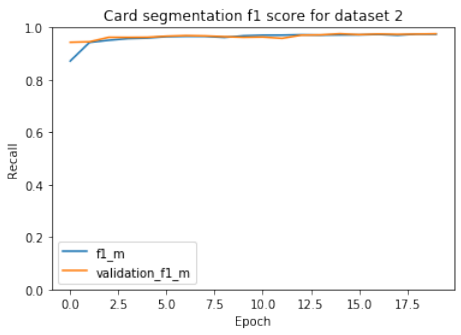
These results were comparable to that of the 5-layer multilabel. This was worse than our expectations as we thought this would improve the f1 score substantially. However, it is likely that with a larger dataset and longer training time, that this model would perform better. This is because this model had over 100 times as many parameters, which would require substantially more effort to train. The 30 million parameters also contributed to the model overfitting, with variance greatly increasing. We should note that even with only fifteen epochs, we were already seeing signs of overfitting with a dropoff in validation f1, while that of the training set was still increasing.

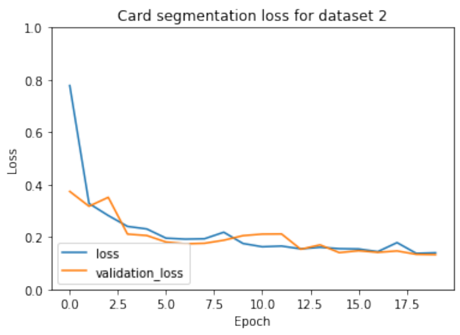
We also considered using an even deeper ResNet with 152 layers, but this did not happen due to constraints in computational resources and budget. We conjecture, however, that this would improve results only marginally given a similar amount of training data and time.



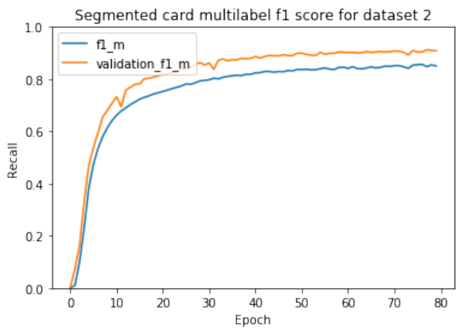


5.5 Image Segmentation





The image segmentation model had very good performance very quickly, achieving 0.976 f1 on both dataset 1 and dataset 2. The segmentation model was trained with adam at 0.001 lr on a dataset of 50,000 images and masks over 20 epochs. This was significantly above our expecations as the original segnet paper has around 0.9 f1 [4][5]. However, we are only using two output mask types whereas SegNet has many mask types.



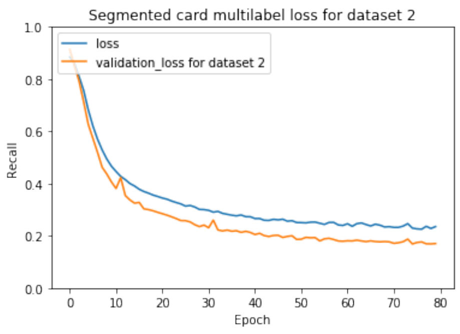


Figure 7: Multilabel results with image segmentation preprocessing on dataset 2

We saw an improved performance with 0.912 f1 score on dataset with segmentation preprocessing. The segmentation model removes the noise of the background and allows the classification model to focus on identifying the cards. However, we saw a worse performance with only 0.765 f1 on dataset 1. This was not expected, but adding the background mask may have removed important information needed for classification.

The multilabel model should learn to identify the foreground itself, but preprocessing enforces that background must be removed. With segmentation, we performed better at classifying images under brightness and contrast variations than previous probabilistic methods [1] with a f1 of 0.6. We believe that CNNs do a better job of removing noise from random variations than probabilistic methods. We believe that our dataset is also more difficult since we use multiple noisy decks.

6 Discussion and Analysis

6.1 Performance

Overall, the performance of the model exceeded our expectations. We were able to get comparable results to the probabilistic methods of Pimentel and Bernadino [1] for images without noise and even beat them for images with noise with a f1 score of 0.912. The multilabel classification model was able to capture different parts of cards even with card rotation, heavy occlusion, various backgrounds, and random lighting to classify the cards.

Our ResNet method showed that a deeper network can provide better results. Our image segmentation method was successful in separating the cards from the background and showed improved performance on the realistic dataset. The autoencoder segmentation model gave us results of 0.976 f1, which was comparable to the around 0.9 f1 score from SegNet [4][5].

We were glad that our method was able to achieve good results on multiple decks and multiple lighting conditions. It gave us confidence that our algorithm could generalize to recognizing cards in the real world.

6.2 Limitations

A major limitation of our model is that we trained our model on images of up to 5 cards. In practice, there can be any number of cards in an image. This is due to time to generate datasets with more cards and infrastructure constraints of handling larger image sizes. However, we believe our methods can be scaled up with more data and more layers to work on image with more than five cards.

Another limitation is that our model is not trained on real images people take of cards. Having a programtically generated dataset is convenient for getting labels for classification or segmentation quickly, but do not capture the nuances of real images. Training on a labeled dataset with real images would help it generalize to real applications.

Compute was an additional limitation to our project. It took around 20 minutes to train one epoch of our 50 layer ResNet on a GPU. With more resources we could train a deep ResNet and use image segmentation preprocessing with ResNet.

6.3 Insights

We learned alot about the computer vision space working on this project. Dealing with problems like arbitrary occlusion of cards, random rotation of cards, noisy backgrounds and changes in brightness taught us how difficult vision problems are. However, the power of CNNs to understand image data with a low number of parameters represents a powerful and flexible tool to build classification models and autoencoders with.

In addition, we learned how to address overfitting with the bias variance tradeoff. Reducing the number of parameters in our models significantly reduced overfitting. Being able to generate data for our model to increase our dataset was also powerful in reducing overfitting.

References

[1] Alexandre Bernardino Joao Pimentel. A comparison of methods for detection and recognition of playing cards. 2012.

[2] Jesper Westell Matias Castillo, Benjamin Goeing. Computer vision for card games http://cs229.stanford.edu/proj2017/final-reports/5233806.pdf. 2017.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[4] Alex Kendall, Vijay Badrinarayanan, , and Roberto Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*, 2015.

[5] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.