Project Progress Report:
Converting Handwritten Mathematical Expressions into LaTeX Representations

APS360
July 12, 2020

Team 41
Neha Basra
Ellen Chen
Ruqhia Frozaan
Cassey Shao

Brief Project Description

Handwritten Mathematical Expressions Recognition (HMER) is the field of converting handwritten mathematical expressions (Figure 1) to their LaTeX representation (Figure 2), which is the goal of our project. This project is useful since scientific documents have been handwritten for centuries (Figure 3), but currently the main source of communication is the internet; therefore it is important to have a tool that can convert expressions[1]. Converting expressions (such as the one seen in Figure 1) manually is tedious, error-prone, and time-consuming[2]. Machine learning can reduce the time and error associated with converting expressions manually. Machine learning is an appropriate tool because we need to classify several characters, which can be done by training a deep network with a large dataset of labelled characters. Also, object detection algorithms like YOLO can be used for image segmentation (Figure 4) to extract single characters.

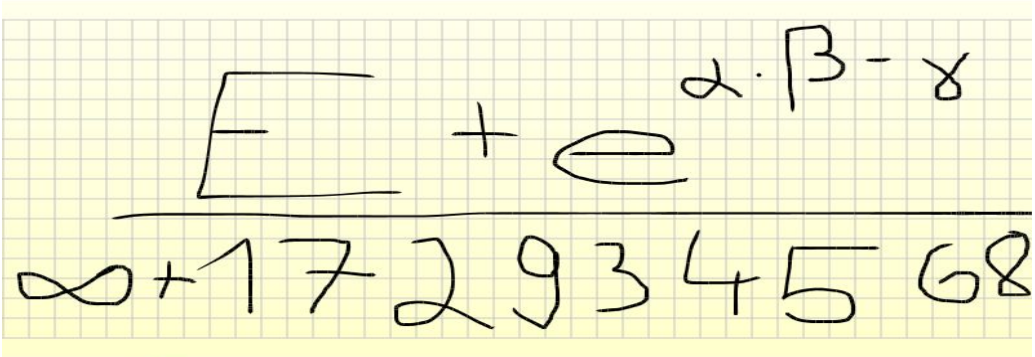


Figure 1: Handwritten mathematical expression.

```
\dfrac {E+e^{\alpha \beta -\gamma }} {\infty +172934568}
```

Figure 2: Latex form of the equation in Figure 1.

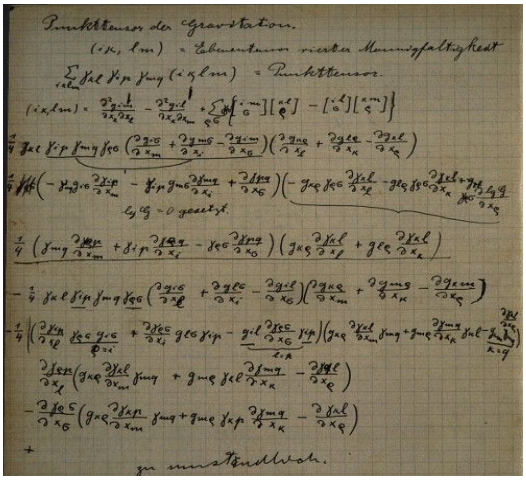


Figure 3: Mathematical document written by Einstein.

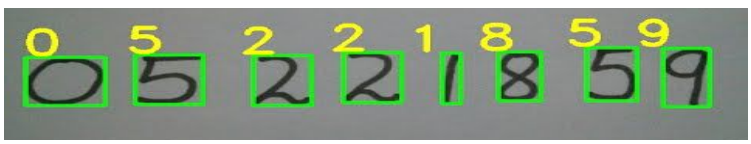


Figure 4: YOLO algorithm used to segment and identify the handwritten digits.

Data Processing

The 3 datasets that we used are: Handwritten Math Expression Dataset[3], English Alphabet Dataset[4] and Hasyv2 Dataset[5]. Below are the steps we took to clean and process the data.

Table of Data Processing Steps

Step #	Description
1.	The dataset that we used to obtain 50 of our classes is the Handwritten Math Expression Dataset. It had 101 classes but these classes were greatly unbalanced as seen from Figure 5.
2.	To balance the dataset we parsed the Handwritten Math Expression Dataset and obtained 100 images of each class using a Python script to automate the process of deleting images. This automated process only took approximately 15 seconds per class. Also, we removed cos, sin, tan, log, and lim classes since the model will train on each letter separately, and hence it will detect each letter separately.
3.	The Handwritten Math Expression Dataset had its capital and lowercase English letters mixed together. Due to the inefficiency of separating them, the letter classes were deleted from this dataset. Instead, we utilized the English Alphabet Dataset from Kaggle.
4.	The letters from the English Alphabet Dataset needed to be inverted in color and resized from 36 by 36 pixels to 45 to 45 pixels as seen in Figure 6. To make processes more efficient, we wrote a Python script that automated the process of inverting colours and resizing for us. This automated process only took approximately 45 seconds per class.
5.	Some letters, such as x, y and z, have capital and lowercase letters that look extremely similar. In hopes of increasing the model's accuracy, we decided to eliminate the capital versions of these letters, and keep the lowercase versions.
6.	The *exists*, *in*, and *forall* symbols in the Handwritten Math Expression Dataset did not have enough samples; therefore, we used the Hasyv2 Dataset from Kaggle to obtain additional samples for these symbols. We used the automation script from step 4 to resize the images from Hasyv2 to 45 by 45 pixels. The distribution of our data is shown in Figure 7.
7.	We have split our data by using 80% for training, 10% for validation and 10% for testing. We can obtain more data if needed from the datasets mentioned above, as they contain many samples for each class.

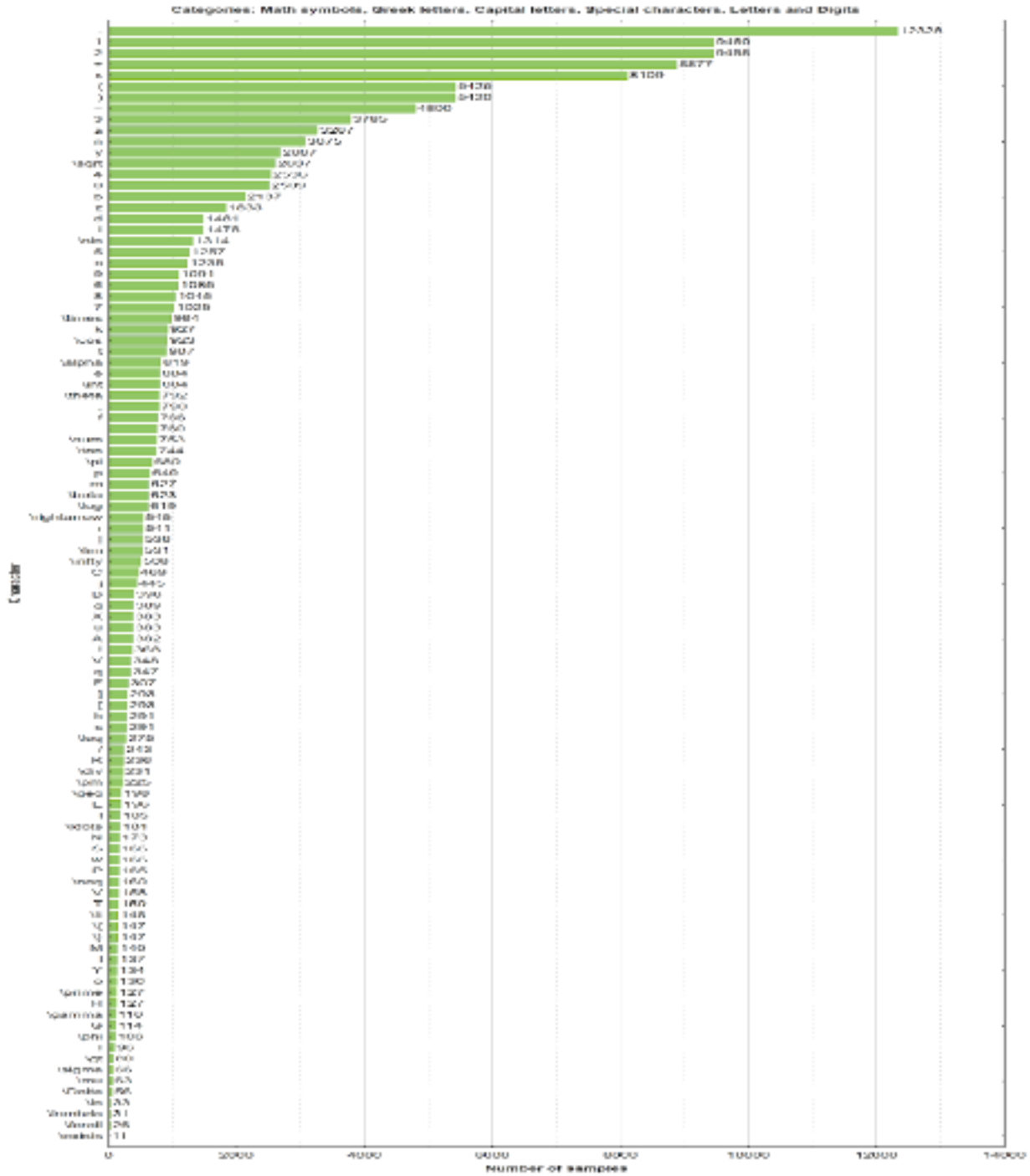


Figure 5: Distribution of classes in the Handwritten Math Expression Dataset[3].

Figure 6: Picture on the left is the original image, and picture on the right is the "cleaned" image after being inverted and resized.

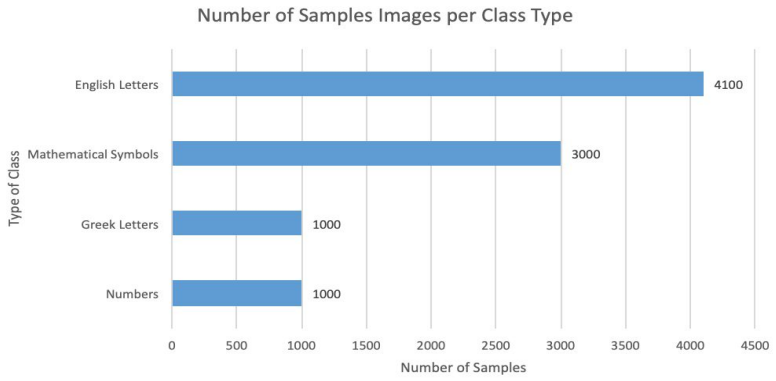


Figure 7: Distribution of *types* of classes in final dataset. Each specific class contains 100 samples.

English letters: includes all letters (capital and lowercase) from A-Z, with only lowercase versions for c, k, o, p, s, u, v, w, x, y and z for reasons explained in step 5 of *Data Processing*.
Mathematical symbols: - ( ) ! [ ] { } + = | ÷ ∫ ∈ / > < ≤ ≥ ∀ ∋ … ∞ ± x Σ ‘ → √
Greek Letters: α β Δ γ λ μ ϕ π Σ θ
Numbers: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Baseline Model

We used multi-classification logistic regression for our baseline model. To implement this, we used Scikit-learn–a Python machine learning library–and imported the logistic regression class. The model was trained/fitted using a sample of our training dataset, and validated using a sample of our validation dataset. This model was a reasonable choice for a baseline model because our project calls for classifying handwritten mathematical symbols (e.g., digits and expression signs); logistic regression is commonly used for classifying handwritten numbers 0 to 9 in machine learning, as seen in [6,7]. Further, the baseline model is easy to implement, requires minimal tuning, and is quick to test.

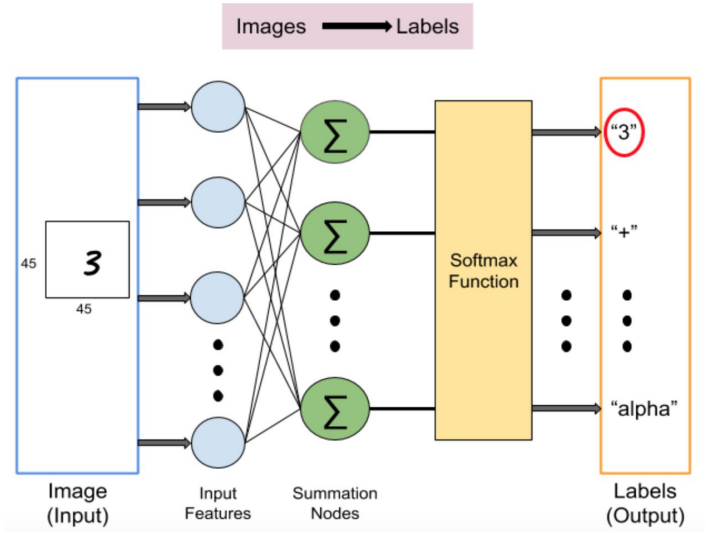


Figure 8: A schematic of the multi-classification logistic regression model. Input features are detected, scaled, summed, then passed through a softmax function to normalize the node output values. These values are then used to predict the output class [8]. Diagram was adapted from [8].

```
# score method gives accuracy of model
# accuracy here is: correct predictions / total number of data points
score = logistic_regression.score(x_test, y_test)
print(score)
```

```
0.3
```

```
from sklearn import metrics
accuracy = metrics.accuracy_score(y_test, predictions)
print(accuracy)
```

```
0.3
```

Figure 9: The validation accuracies for the initial implementation of the baseline model.

Primary Model

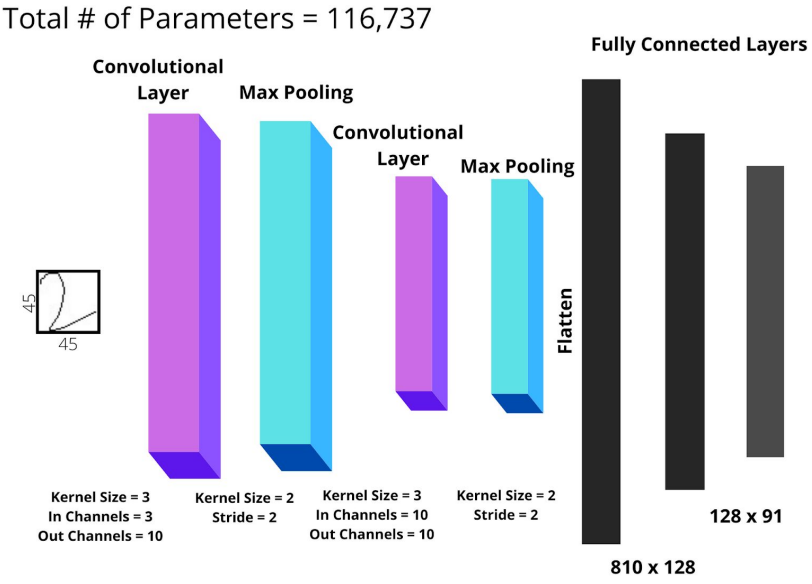


Figure 10: Diagram of primary CNN model architecture with 2 convolutional layers with max pooling in between, with 3 fully connected layers. It utilizes the ReLU function and the Adam optimizer. Due to our project being based on image classification of symbols (numbers and alphabet characters), we decided to use a convolutional neural network (CNN). The CNN model is our primary model due to the fact that our CNN model gained higher accuracy than both of the transfer models after fine tuning hyperparameters. We will continue to experiment with fine-tuning the CNN model with transfer learning to increase accuracy.

Table of Hyperparameter Values for Primary Model

Hyperparameter	Value
Epochs	15
Batch Size	64
Weight Decay	0.0001
Learning Rate	0.001

Results

Primary Model Results

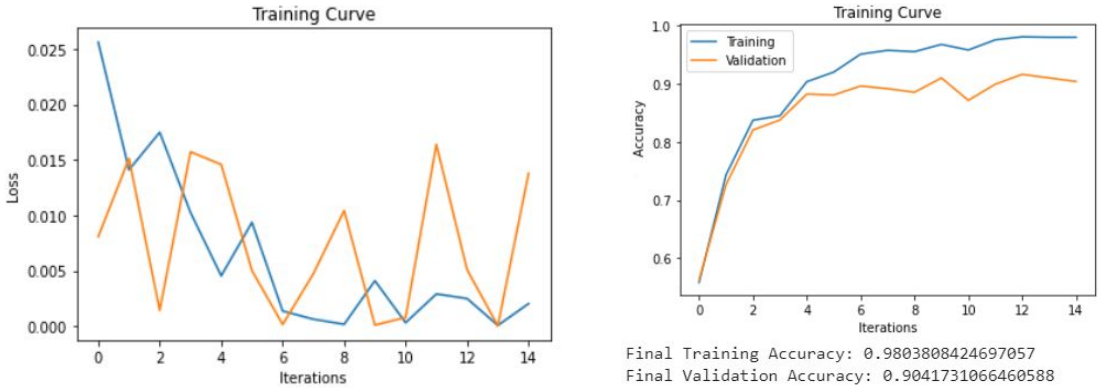


Figure 11: Accuracies of our primary model (CNN): Training Accuracy of 98.04%, and Validation Accuracy of 90.42%

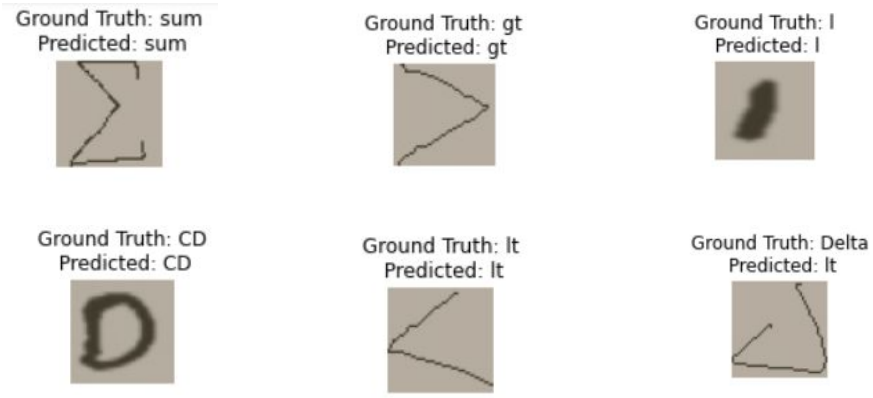


Figure 12: Comparisons from ground truth and our model's predictions, along with pictures of the dataset. We will be analyzing this data to determine how to make our model more robust.

ResNet Results

We utilized ResNet50 as a fixed feature extractor, where we kept all the weights except for the last layer which is the only layer that is trained, and it is shown in Figure 13.

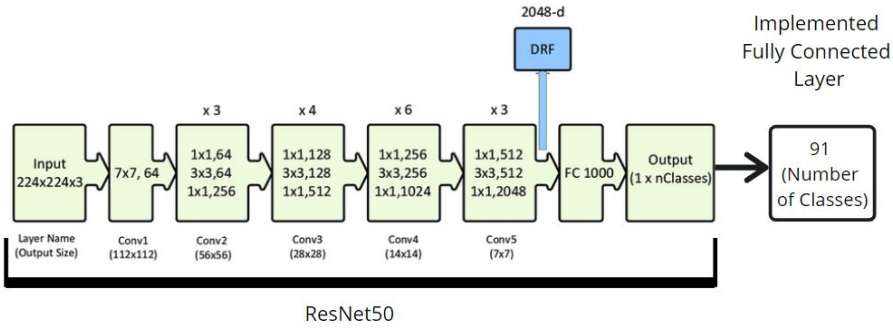
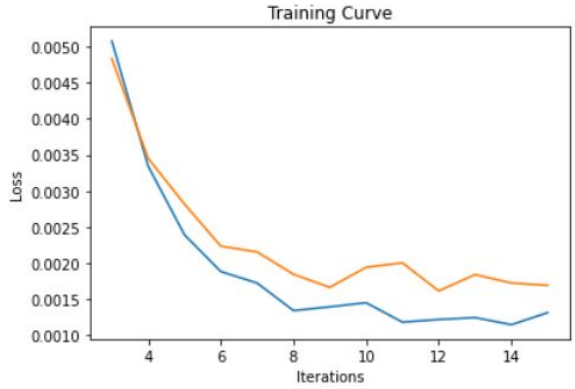
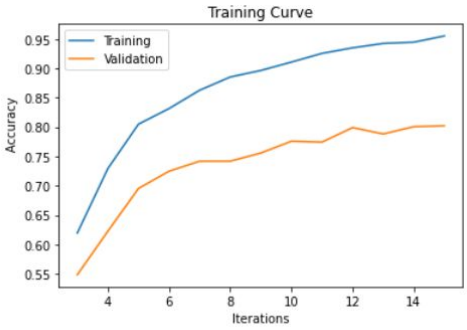


Figure 13: The model architecture of our transfer learning with fixed feature extraction with ResNet50, figure modified from [9].

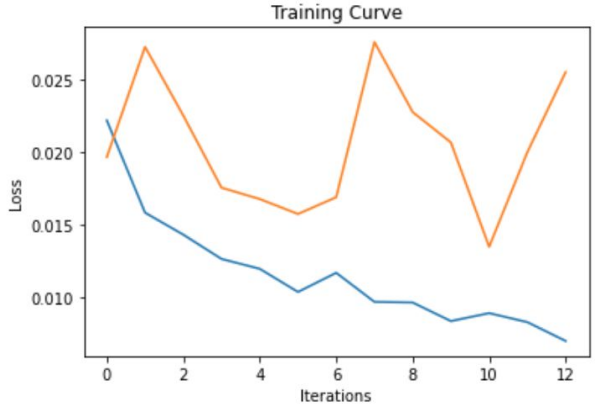


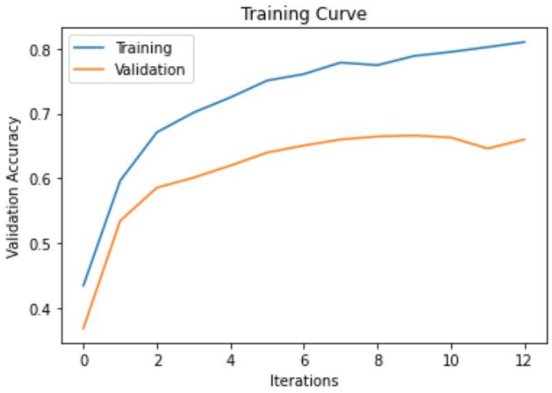



```
Final Training Accuracy: 0.9517214849009424
Final Validation Accuracy: 0.8021638330757341
```

Figure 14: Loss and Accuracy Graphs for Transfer Learning with ResNet50: Training Accuracy of 95.17% and Validation Accuracy of 80.22%.

GoogLeNet Results






```
Final Training Accuracy: 0.8099634545104828
Final Validation Accuracy: 0.6599690880989181
Total time:   1134.43 s  Time per Epoch:  87.26 s
```

Figure 15: Loss and Accuracy Graphs for Transfer Learning with GoogLeNet: Training Accuracy of 81% and Validation Accuracy of 67%.

Table of Hyperparameter Values for Best Transfer Learning Models

Hyperparameter	Value for ResNet	Value for GoogLeNet
Epochs	13	20
Batch Size	512	256
Weight Decay	0.001	0.001
Learning Rate	0.001	0.001

Project Progress

Table of Team Communications, Project Management, Contributions, and Overall Progress

How Team is Working Together	Team Communications: Shared Google Drive/Docs/Colab (shared code), Messenger, Audio Calls/Regular Meetings.
Project Management	Track progress: We have a worklog on Excel where each task throughout the course of the project is listed and assigned to specific members; each member records the date they completed their assigned task. Track results: All graphs and accuracies are screenshotted and recorded on a separate Google Docs page with each member's name as the heading. This helps the team keep track of hyperparameters and model results that each member was able to achieve. Track updates to shared code: Each member comments out their names at the top of each code block they used. Also, the "Comments" tool is regularly used to ask and answer questions on team members' codes.
Each Member's Contributions	**Ruqhia**: researched and combined datasets; performed image segmentation using OpenCV, split and loaded the data; tuned hyperparameters. **Neha**: researched and combined datasets; wrote python script to adjust number of images per class; tuned hyperparameters. **Cassey**: implemented GoogLeNet and tuned it; researched, implemented, and tested data on the baseline model; tuned hyperparameters. **Ellen**: implemented ResNet and tuned it; constructed CNN model; tuned hyperparameters; visualized the data.
Progress	Each member completed assigned tasks from the Project Proposal by the deadlines.

Table of Updated Project Plan and Distribution of Work

Week/ Deadline	Task	Assigned Members
Week 9 (after Progress Report)	Improving Model ● Continue to tune hyperparameters. ● Research different transfer learning techniques.	Ellen, Cassey
	Object Detection ● Research and implement YOLO for object detection.	Neha, Ruqhia *Redundancy: If Neha and Ruqhia require assistance, Ellen and Cassey can offer help with the YOLO implementation (after finishing their assigned task above).

Week 10	Evaluate the Model ● Evaluate model on *testing* set. Improve and Finalize Model	Neha, Ellen All Members *Redundancy: If model's results are not desirable: dataset can be shortened down, or we can limit our scope by only testing on images that do not have exponents/subscripts/superscripts
Week 11	Videography ● Create a detailed video plan. Final Deliverable ● Write a rough draft of the final report.	Ruqhia, Ellen All Members *Redundancy: If a team member is unable to contribute enough this week (due to personal reasons), that member can be re-assigned more work on the video/report for week 12.
Week 12	Videography ● Edit using iMovie. Submit video. Final Deliverable ● Edit rough draft and prepare the final copy of the final report.	Neha, Cassey All Members
Week 13	Present video and submit final report.	All Members

References

[1] A.-M. Awal, H. Mouchère, and C. Viard-Gaudin, "Towards Handwritten Mathematical Expression Recognition," *2009 10th International Conference on Document Analysis and Recognition*, Jul. 2009.

[2] A. Schechter, *Pdfs.semanticscholar.org*, 2017. [Online]. Available: https://pdfs.semanticscholar.org/fd25/9c0d70368a50b52755d4e33590566ef5d373.pdf?_ga=2.12415604.1536964691.1591985627-78053362.1591985627. [Accessed: 12- July- 2020].

[3] X. Nano, "Handwritten math symbols dataset," *Kaggle*, 15-Jan-2017. [Online]. Available: https://www.kaggle.com/xainano/handwrittenmathsymbols. [Accessed: 12-Jul-2020].

[4] Mohneesh_Sreegirisetty, "English Alphabets," *Kaggle*, 18-Jul-2018. [Online]. Available: https://www.kaggle.com/mohneesh7/english-alphabets. [Accessed: 12-Jul-2020].

[5] Solo, "HASYV2," *Kaggle*, 10-Apr-2019. [Online]. Available: https://www.kaggle.com/guru001/hasyv2?fbclid=IwAR3001xLNRRzVPyb7xjYjWYDqzW_RT. [Accessed: 12-Jul-2020].

[6] M. Galarnyk, "Logistic Regression using Python (scikit-learn)," *Medium*, 29-Apr-2020. [Online]. Available: https://towardsdatascience.com/logistic-regression-using-python-sklearn-numpy-mnist-handwriting-recognition-matplotlib-a6b31e2b166a. [Accessed: 12-Jul-2020].

[7] mGalarnyk, "mGalarnyk/Python_Tutorials," *GitHub*, 03-Nov-2017. [Online]. Available: https://github.com/mGalarnyk/Python_Tutorials/blob/master/Sklearn/Logistic_Regression/LogisticRegression_toy_digits_Codementor.ipynb. [Accessed: 12-Jul-2020].

[8] Roland, "Deep Learning Explained: Logistic Regression," *Microsoft Azure Notebooks*. [Online]. Available: https://notebooks.azure.com/rfernand/projects/edxdle/html/Lab2_LogisticRegression.ipynb. [Accessed: 12-Jul-2020].

[9] A. Mahmood and A. Giraldo, *ResNet-50 architecture shown with the residual units, the size of the filters and the outputs of each convolutional layer.* ResearchGate GmbH, 2019.