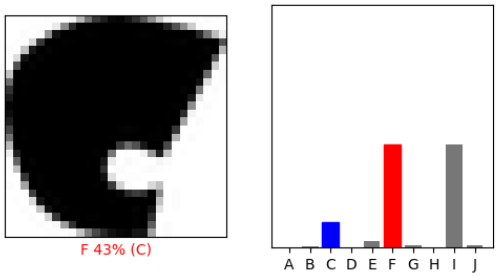
CPSC 433 Assignment 1

Ali Akbari, 30171539

Part 1:

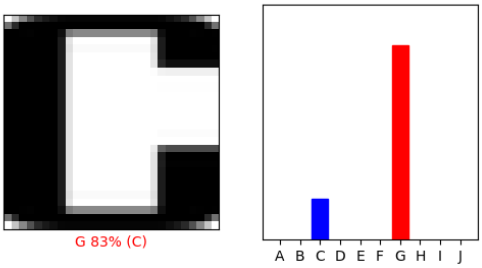
1. The initial notMNIST-Partial model was a basic neural network with the following architecture: Flatten layer: Converts the 28x28 pixel grayscale images into a 1D array of 784 values. Dense layer with 128 units: Uses ReLU activation for non-linearity. Dense layer with 64 units: Again uses ReLU activation for added complexity. Dense output layer with 10 units: Represents the 10 classes ('A' through 'J') using a softmax activation function for probabilistic outputs. The model was compiled using the SGD optimizer, sparse categorical crossentropy as the loss function, and accuracy as the evaluation metric. Training was conducted for 20 epochs, achieving: Training accuracy: 91.0% Test accuracy: 93.2% While the model performed decently on the test set, it struggled to generalize well on more challenging examples. The observed gap between training and test accuracy suggests minor overfitting.

2. **Image 1:**



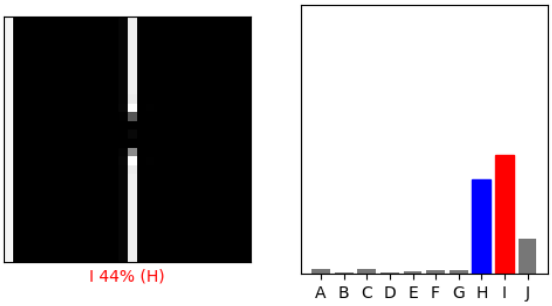
True Label: C
Predicted Label: F

Image 2:



True Label: C
Predicted Label: G

Image 3:



True Label: H
Predicted Label: J

3. The changes made to transition from the Partial model to the Complete model significantly enhanced its performance by addressing the limitations of the earlier design. The Complete model introduced Convolutional Neural Network (CNN) layers. Specifically, the Complete model included two Conv2D layers with ReLU activation, which allow it to learn hierarchical features from the images, starting with low-level features (e.g., edges) and progressing to more complex patterns (e.g., shapes). Each convolutional layer was followed by Batch Normalization, which stabilizes and accelerates training by normalizing intermediate outputs, and MaxPooling, which reduces spatial dimensions while retaining important features. To combat overfitting, Dropout layers were added after both convolutional and dense layers. These layers randomly disable a fraction of neurons during training, forcing the network to learn more generalized features rather than memorizing the training data. Additionally, Adam optimizer replaced SGD, offering adaptive learning rates for faster convergence and improved performance. Finally, the inclusion of additional Dense layers with Batch Normalization and Dropout in the Fully Connected Network ensured that the final features extracted by the CNN layers were effectively utilized for classification. Together, these changes allowed the Complete model to achieve better generalization, as evidenced by improved test accuracy and its ability to correctly classify challenging examples.

4. **Changes to predict_test.py:**
   To make the predict_test.py files functional with the trained model, several modifications were made. First, the line that originally set the model to None was updated to load the trained model from the specified file using tf.keras.models.load_model(sys.argv[2]). This allows the model to be loaded. Second, the placeholder array for the prediction was replaced with prediction = model.predict(img) and prediction = prediction[0], which generates the confidence array for each class based on the input image. This array contains the model's predicted probabilities for each class. Finally, the line that previously set predicted_label to a fixed value of 0 was replaced with predicted_label = np.argmax(prediction), which identifies the class with the highest probability by finding the index of the maximum value in the prediction array. These changes enable the program to make predictions with the loaded model and visualize the results by displaying the predicted and true labels alongside the input image.

Part 2:

1. The dataset was split into training and testing sets with approximately 6/7 of the data used for training and 1/7 for testing. This ratio ensures that the model has a sufficient amount of data to learn patterns while reserving enough unseen data to evaluate its performance. The split was performed manually, ensuring randomness in the selection of training and test samples to avoid bias. By maintaining this proportional split, the training set is representative of the data, while the test set is large enough to provide a meaningful assessment of model generalization.

2. The "position" column contains categories like 'RB' (Running Back) and 'QB' (Quarterback). To handle this, each position was first given a unique number using a mapping. Then, these numbers were converted into binary format using one-hot encoding. This way, the model can understand the positions as separate categories without treating them like numbers that can be ranked or compared.

3. To mitigate overfitting, the model includes dropout layers with a rate of 0.5, which randomly deactivates neurons during training to reduce dependency on specific features. Batch normalization was applied after each dense layer to stabilize and accelerate the learning process while reducing sensitivity to initializations. The number of epochs was kept relatively low (20) to avoid overtraining, and the model's performance on the validation set (test data) was monitored during training. These techniques collectively improved the model's ability to generalize to unseen data.

4. Model Architecture: The model includes two dense layers with 512 and 256 neurons, respectively, using ReLU activation, followed by a final output layer with a sigmoid activation for binary classification.

   Dropout Rate: 0.5 for each dropout layer.
   Batch Size: 16.
   Epochs: 20.
   Optimizer: Adam.
   Loss Function: Binary cross-entropy.
   Normalization: Input features were standardized to have zero mean and unit variance.

   Results:
   Training Accuracy: ~ 82%
   Test Accuracy: ~ 70%