Classification of Sketches: Quick, Draw! Doodle Recognition Challenge

By: Group 10 The Sketchy Guys
Team Members:

1. Rajdeep Adak (1230059612)
2. Prajjwal Dutta (1229914740)
3. Vivek Kulkarni (1228844827)
4. Varshini Rao (1229838274)
5. Harshavardhana Chadaram (1230711952)

Problem Definition

Develop an improved classifier for the Quick Draw! dataset, which contains 50 million drawings across 340 label categories. The challenge lies in creating a model robust enough to handle noisy and incomplete drawings while generalizing effectively to a manually labeled test set.

Impact

Success in this task not only advances pattern recognition solutions but also holds practical implications for OCR, ASR, NLP, and related fields. Improved classifiers enable more accurate interpretation of handwritten text and drawings, enhancing accessibility and efficiency in tasks such as document digitization and language processing.

Approach

Our method involves preprocessing doodle images, followed by constructing a CNN model. Training the model involves optimizing parameters with Adam optimizer and monitoring performance on training and test datasets. This comprehensive pipeline ensures robust doodle recognition with potential applications in various domains.

Methodology

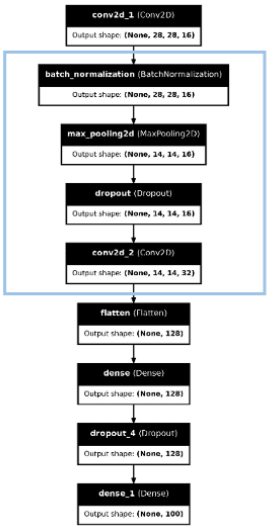
a. Data preparation:

In the data preparation phase of our doodle image recognition project, we opted to utilize Numpy bitmaps instead of CSV files for processing efficiency and ease of manipulation. Numpy provides a powerful array object that makes data manipulation computationally efficient. The choice to use bitmap format allows us to handle images in a binary representation, where each pixel is denoted as either on or off, which simplifies the structure of the data and reduces file size. This binary format is especially advantageous for doodle images, which typically do not require the full color spectrum or grayscale gradients.

b. Architecture of the CNN:

Our convolutional neural network structure is designed with multiple layers that play pivotal roles in pattern recognition and image classification. The core of the network consists of convolutional layers responsible for extracting features, accompanied by pooling layers that downsample the image to reduce dimensionality. The ReLU activation functions inject non-linearity into the processing, which is essential for capturing complex patterns. The network also includes dense layers that contribute to final decision-

making, dropout layers that serve as a form of regularization to prevent overfitting, and batch normalization layers that maintain the stability of activations throughout the network.

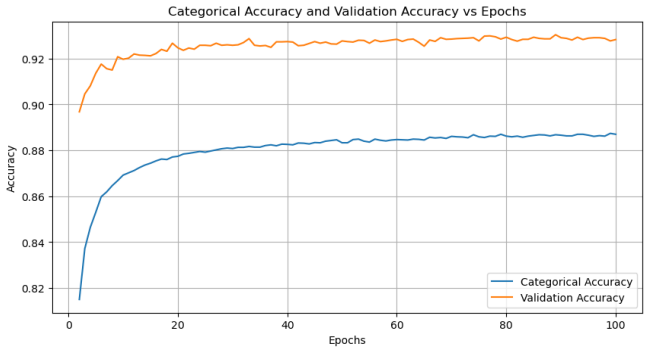


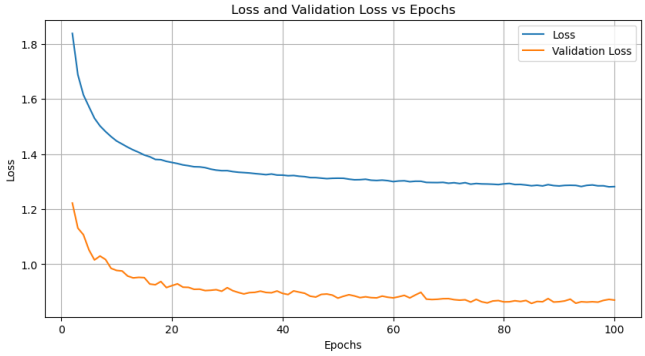
x 4

c. Learning Pipeline

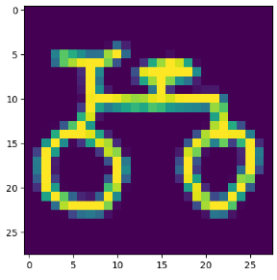
To initiate our experiment, we utilized the load_data function to import the dataset. Preprocessing involved resizing images to a uniform size of 28x28 pixels. We also normalized the pixel values to fall within a 0-1 range to facilitate optimization during the training process. The training procedure for the provided code involves loading and preparing doodle image data, constructing a CNN model with 4 convolutional layers each having batch normalization, max-pooling, dropout, and dense layers, followed by compiling the model with categorical cross-entropy loss and the Adam optimizer. Subsequently, the model is trained on the training data with specified hyperparameters, and its performance is evaluated on both training and test datasets. Inference is then demonstrated by predicting doodle categories for randomly selected test images.

Model Evaluation





Results



In our project, we achieved a test accuracy of 93.17%, ranking 171st out of 1309 submissions in Kaggle. The best results on the leaderboard were attained by CNN-based approaches, with the highest Public score reaching 95.6% and the top (CNN-only) score being 94.5%.

```
Epoch 100/100
4750/4750 - 103s - 22ms/step - loss: 1.2820 - top_k_categorical_accuracy: 0.8870 -
val_loss: 0.8696 - val_top_k_categorical_accuracy: 0.9283
```

How to improve?

- Increasing the number of layers or neurons per hidden layer can enhance the model's ability to learn complex patterns but requires careful tuning to avoid overfitting.
- Scheduling the learning rate can help the model converge faster and potentially reach a better optimum by adjusting the learning rate during training.
- Regularization techniques such as L1 or L2 regularization, dropout, or batch normalization can prevent overfitting by adding penalties to the loss function or introducing noise during training.
- Data augmentation, which involves applying random transformations to the training data, can help the model generalize better to unseen data and improve performance.