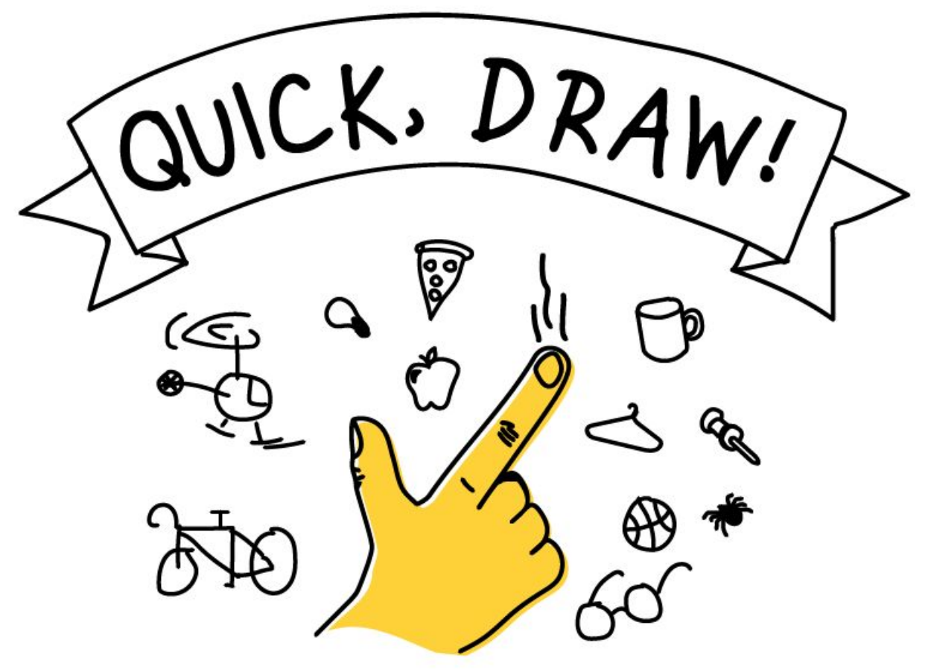
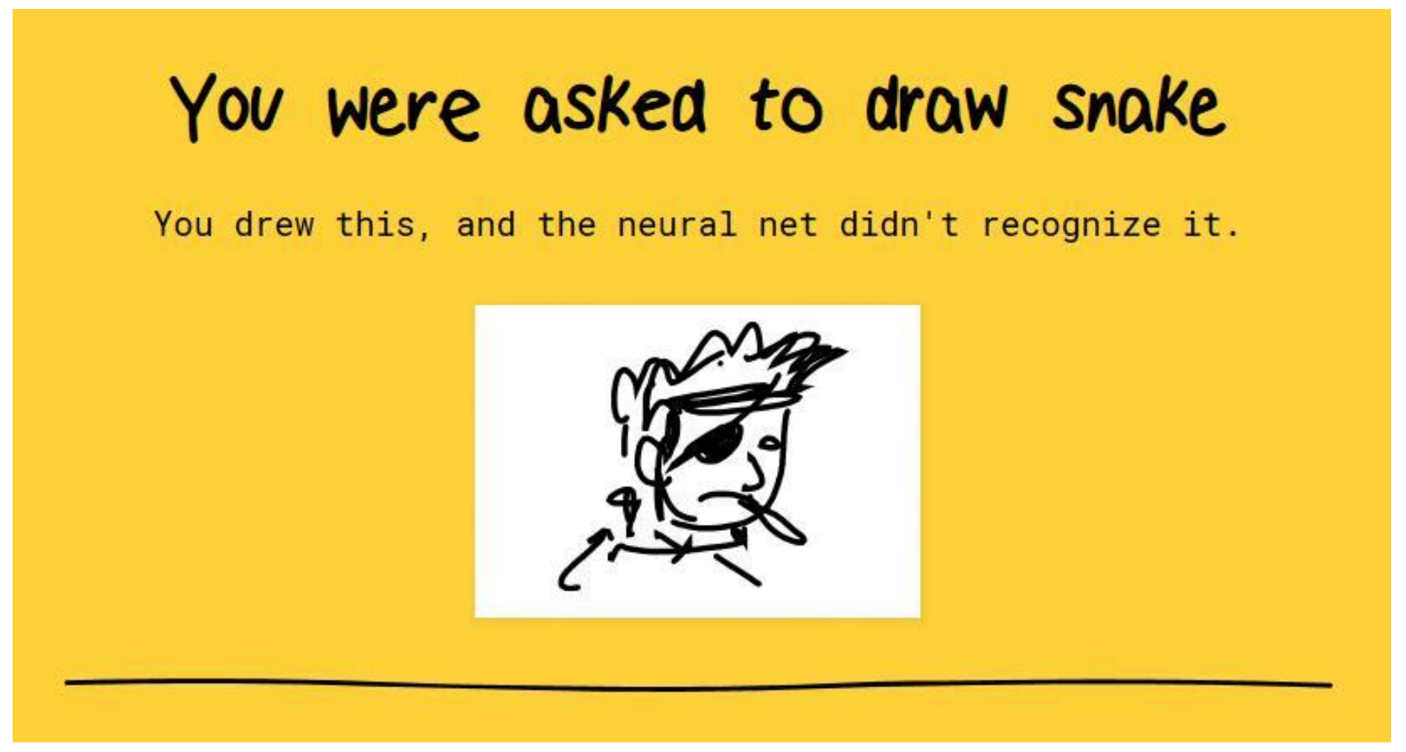


Doodling in the Deep

Google Quick, Draw! Challenge

Ramesh Balaji | Ernest Yucheng Chang | Tanmaya Dabral | Irina Javed


You were asked to draw snake
You drew this, and the neural net didn't recognize it.

Problem Statement

Can a bot play Pictionary? Our system aims to be exactly that. The task is to build a recognition system that, given a doodle by a person, can classify it. The concepts that it needs to guess can vary from simple things like ‘purse’ to complicated ones such as ‘The Mona Lisa’ or ‘animal migration’.

Dataset

The data is from the Google Quick, Draw! challenge. Notes on the dataset:

- Huge. 50 million data points (testing: 112K), 340 categories
- Noisy data. Since the data comes from quick draw game itself, the drawings can be incomplete or may not match the right label.
- Data are represented as a series of strokes.
  - Raw version - each stroke is a list of all the points in the stroke.
  - Simplified version - each stroke is a list of anchor points for an approximate polyline, generated by running the Ramer - Douglas - Peucker algorithm on the raw stroke.

Hypothesis

The nature of the dataset suggests the possibility of two distinct approaches:

- Treating each doodle as an image, exploiting the spatial information.
- Treating each doodle as a time-series of strokes, exploiting the temporal information.

We hypothesize that a combination of the two approaches will yield a better result than either approach alone.

Models and Performance

Conv1D + LSTM-128 (stroke-based)

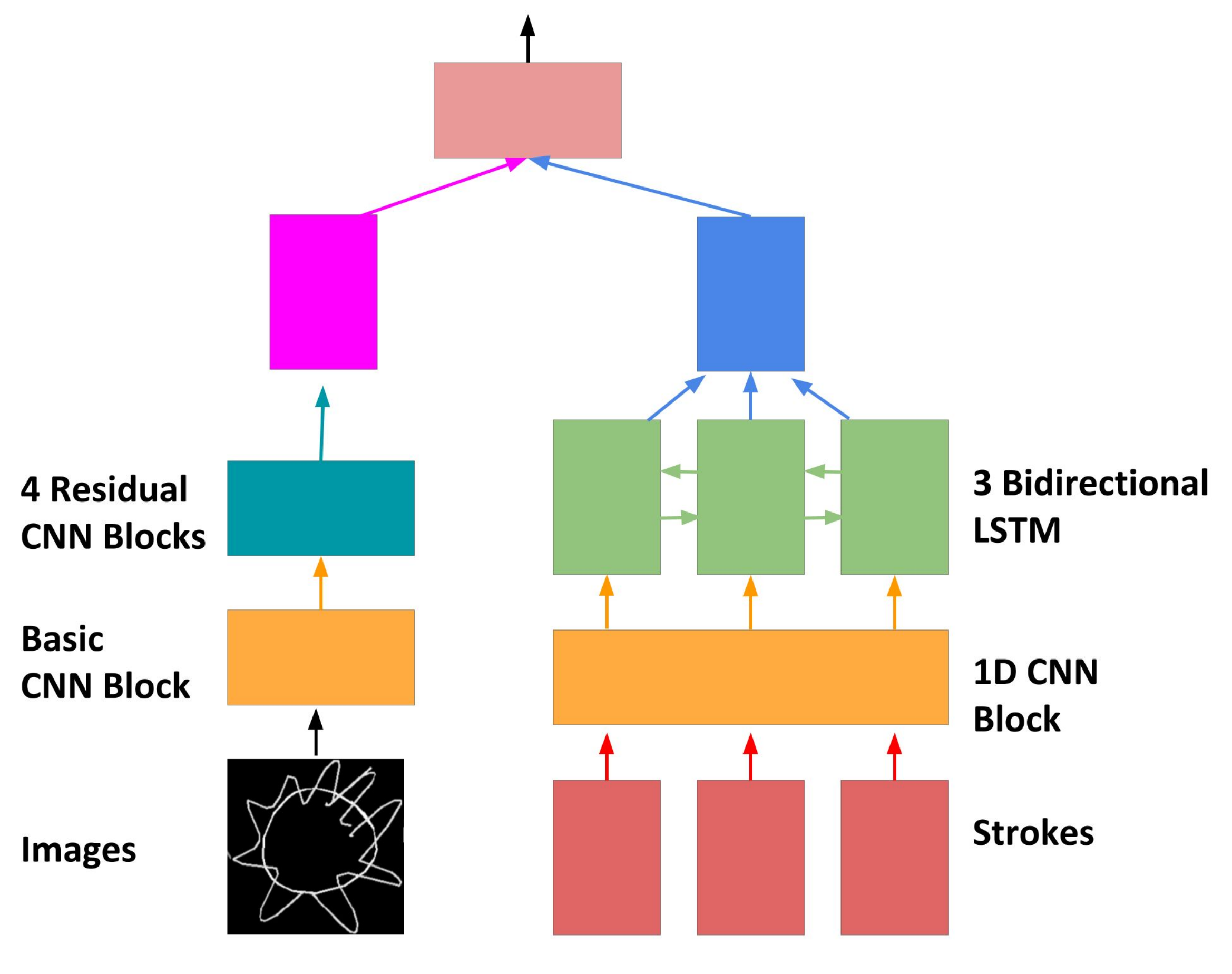
The preprocessed stroke input is passed through 3 1D Convolutional layers to augment the dimensions to (length x 96). They are then fed to a 3-layered LSTM with 128 hidden units, the outputs of which are averaged across timesteps and fed to a linear layer to generate the logits.
Training data used: 100% **MAP@3:** 0.804

Conv1D + BiLSTM-512 (stroke-based)

Similar to the previous LSTM model, with the LSTM replaced with a 3-layered BiLSTM with 512 hidden units.
Training data used: 10% **MAP@3:** 0.864

Time-coded ResNet18 (image-based, 224x224)

The images were generated in a time-coded fashion. A TorchVision ResNet18 was trained over these images.
Training data used: 10% **MAP@3:** 0.889



The ResNet18-BiLSTM ensemble

MobileNet V2 (image-based, 28x28)

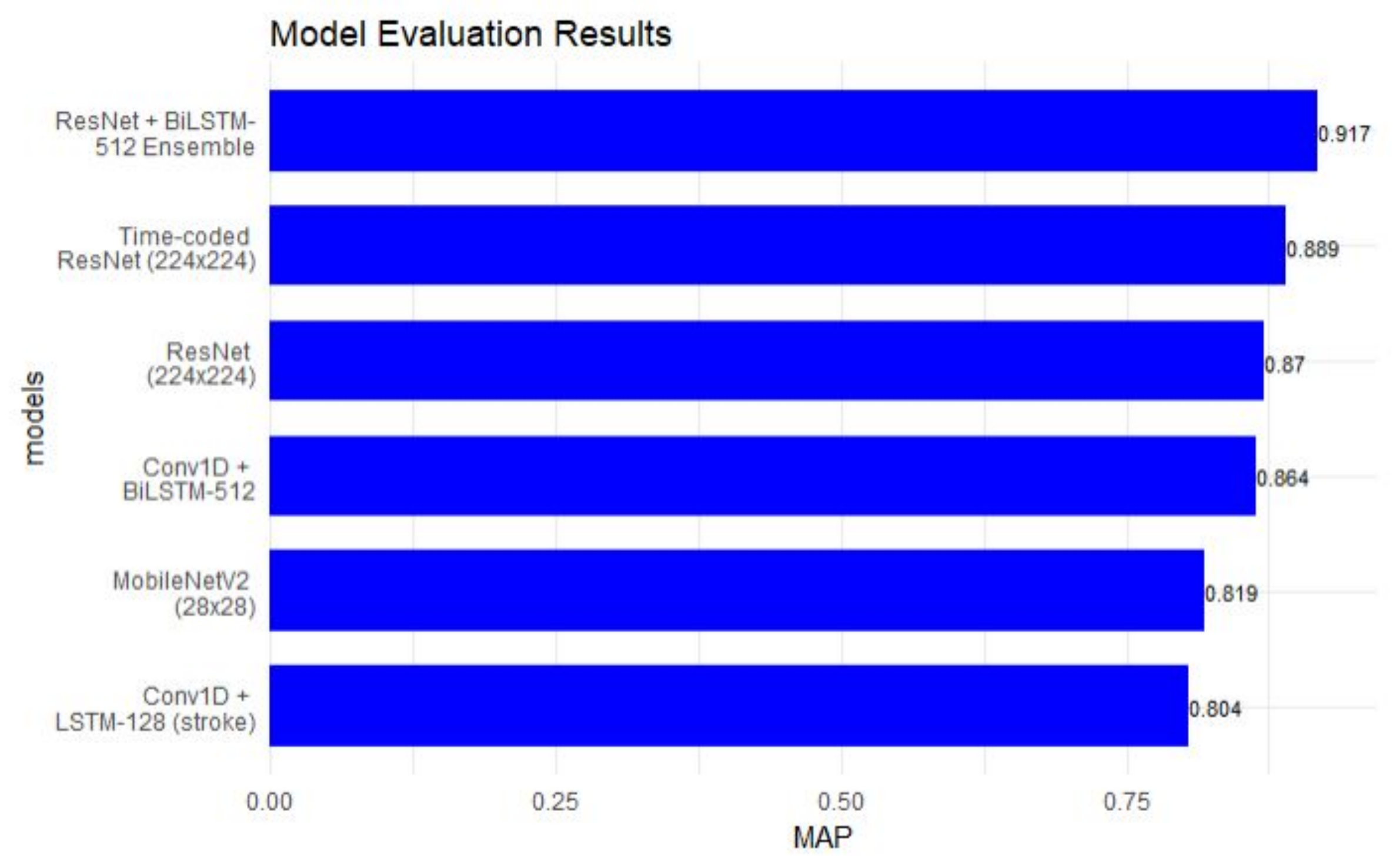
MobileNet V2 begins with a convolution layer with 32 filters, followed by 19 residual bottleneck layers, each of which contain a depthwise separable convolution with inverted residuals. This model was designed to be memory efficient and to successfully balance computation and accuracy.
Training data used: 100% **MAP@3:** 0.819

ResNet18 (image-based, 224x224)

TorchVision’s ResNet18 model, which is a normal convolutional block followed by a stack of four residual convolution blocks.
Training data used: 10% **MAP@3:** 0.870

ResNet18 and Conv1D + BiLSTM-512 (stroke-based) ensemble

The ResNet18 and the Conv1D + BiLSTM-512 (above) were pre-trained on 10% of the data. The models were then combined as in a LogitBoost ensemble, i.e. the final logits is a weighted sum of the logits from the models. The weights & parameters for each model were made trainable.
Training data used: 10% **MAP@3:** 0.917



Data Preprocessing

For sequence of strokes approach:
Concatenate strokes into single sequence. Add a third dimension to each point, 1 if beginning of a new stroke, 0 otherwise. Squish all the coordinates to [0, 1] to provide scale invariance

For image based approach:
Plot points using data with OpenCV polyline to generate images. Image size 224 x 224 for use with ResNets. We also normalize pixel values by dividing 255.

Our improved approach:
Encode order of strokes info by drawing the picture with different line intensities. The assumption is that the first several strokes are more important than strokes drawn later. Helps model learn useful patterns.

Results & Conclusions

With such a large dataset, a single epoch for just 10% dataset could take, depending on the model, half a day to over a single day. Thus our results, given our limited time and resources, demonstrate our collective effort to balance accuracy with efficiency, as well as creative ways to utilize the dataset at hand.

The mean average precision with three guesses (MAP) given by Kaggle for each of our models is shown to the left. Our results are in agreement with our hypothesis in that the ensemble performs best out of all the models. The time-coded Resnet18 performs well too. These are the two models that use both the time component and the spatial component.