CS5242 Project Report - Breakfast Actions Classification

Authors: Amrut Prabhu (A0162655B), Aadyaa Maddi (A0161468Y), Yash Chowdhary (A0162026R)

Abstract

In this paper, we classify the various actions taken when performing common breakfast tasks, given feature embeddings extracted from videos recordings of these tasks. We focus on using a recurrent neural network architecture to meet our goal of high classification accuracy. Our experiments show that a stacked bi-directional GRU is the most suitable model, and gives a test accuracy of 70.482% on Kaggle.

Contributions

Amrut Prabhu: Wrote scripts for training and testing BiRNN, BiLSTM, BiGRU and stacked BiGRU models for the "segment to label" approach; generated visualisation for our results; contributed to relevant sections in the report.
Aadyaa Maddi: Wrote Keras scripts needed for data processing and training/testing models for the "segment to label" approach; trained the LSTM models; made contributions to the report.
Yash Chowdhary: Wrote scripts needed for data processing and training/testing models for the "frame to label" approach; trained the baseline DNN model; made contributions to the report.

1. Introduction

The applications of computer vision encompass an array of real-world problems from the domains of healthcare and military to those of retail and services. Some of the common tasks that these real-world problems can be reduced to include image classification, image segmentation and object detection. Performing these tasks provides ample information about images as it is the key to answering questions such as "What is this?" and "Where is it?". Extending this to the domain of videos is but a matter of performing this on a sequence of images (or frames) - after all, that is what a video is.

In this paper, we seek to perform classification on a dataset of videos that involve common breakfast actions, ranging from 'take cup' and 'pour milk' to 'cut fruit' and 'fry pancake' - which can be portrayed as 'segments' in the videos. We make use of a recurrent neural network architecture composed of Gated Recurrent Units (GRUs) as they provide the benefit of taking in **variable-length input** and because they were built for processing **sequential data** (like the frames of videos) that have an associated temporal element. Our experimental strategy can be broadly classified into 2 approaches - classifying each frame, and classifying action segments of videos. In each approach, we try several combinations of RNN models that are tuned to eventually obtain the maximum accuracy. We discuss and describe the tasks performed for preprocessing data and evaluate the performance of our best model when it comes to classifying these breakfast action segments in videos.

2. Related Work

To carry out our task, we are not provided with the raw videos of the breakfast actions; instead, we are given the feature vectors of the individual frames in a video. These features are derived by using Two-Stream Inflated 3D ConvNets [1] - the frames and the corresponding optical flows are passed through 3D ConvNets and concatenated. The resulting output is a 400-dimensional I3D feature vector for each frame.

3. Methodology

3.1. Problem Statements

A. Segment to Label: The model takes in either a segment alone OR the entire video and segment indices as input and predicts labels for each segment.
B. Frame to Label (**Bonus**): The model takes in one video (i.e. a list of frames) as the input and predicts a label each frame in the video.

3.2. Dataset

To tackle these two problem statements, we perform a descriptive analysis of the videos and the segments of the videos. The dataset consists of features of frames from 1712 videos - 1460 of which are for training. We perform inference on the remaining 252 of them, and generate a CSV file that maps the action segments to an action label. For both problems, we further partition our training dataset into training and validation using an 80-20 split.

	video_length
count	1460
mean	2113.340411
std	1846.543551
min	130
25%	785
50%	1373
75%	3021
max	9741

	seg_len
count	5660.000000
mean	407.330389
std	554.839000
min	8.000000
25%	133.000000
50%	235.000000
75%	432.000000
max	5791.000000

Figure 1: Numerical Analysis (a) Video Lengths (b) Segment Lengths

For the "Segment to Label" problem, we have 5660 segments for training, as can be seen in the numerical analysis in Figure 1 (b).

The labels which we need to classify our action segments into are 48 in number. However, the label corresponding to 'Silence' (SIL) in the frames is disregarded as we only want to learn from frames that have an action. We thus ignore the segments that are labeled with the '0' ground truth value for the "Segment to Label" problem. We do the same for the

"Frame to Label" problem, trimming the videos at the beginning and the end - where the 'SIL' frames are present.

Figure 2: Distribution of Classification Labels

We also observe that the bulk of the segments have a length of 200-400 frames, as can be seen in Figure 3.

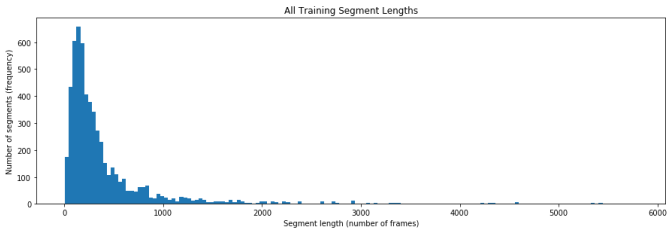

Figure 3: Training Segment Lengths Histogram

Similarly, we also observe that the median length of the videos (number of frames) is around the 1400 mark.

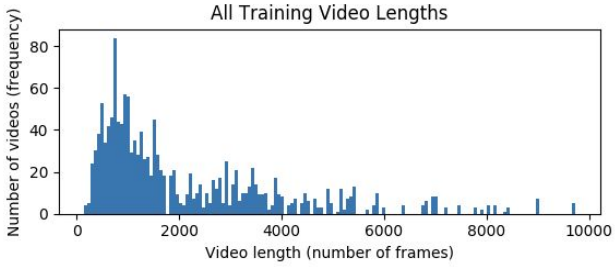

Figure 4: Training Video Lengths Histogram

3.3. Data Processing

Segment to Label Data Processing

For this problem, a record in the dataset is a segment and its corresponding label. We extract segments from each video using the segment split data and encode the label for each segment by taking a one-hot representation of the first frame label of that segment. We also pad each segment to length 6000, as the maximum possible length of a segment in the given dataset is **5791**. Note that the segments do not retain any information about the videos they are in.

Frame to Label Data Processing

For this problem, a record in the dataset is the features of all the frames of a video and a list of labels corresponding to each frame. We encode each label in the list using a one-hot representation. We also pad each video to length 8500, despite the maximum possible length of a video in the given dataset being **9741**. The rationale behind this is that the bulk of the videos have a length in [1000, 3000], and there are very few videos above the length of 8500. Since we are going to divide these records into batches for training, we wish to minimize padding. Hence, we choose to forgo information after the 8500th frame for lesser padding.

Finally, for the "Segment to Label" file, we create a training segments file to partition the segments within a video which follows the same format as the test segments file. We don't require such a file for the "Frame to Label" approach as the granularity is reduced to frames. For each approach, we rely on the test segments file *solely* for inference i.e. to know where to partition the segments while creating the CSV file with the test results.

3.4. Classifier Models

Dense Neural Network

We train a dense neural network (**model B1**) to establish a baseline for the "Frame to Label" problem statement. The model takes in the I3D features of the video frames as mentioned in the previous section, and passes it through several dense layers. The model uses regularisation techniques such as batch normalisation (BN) and dropout layers (with *p=0.5*). The entire model architecture can be viewed in the appendix, for clarity. The model is trained for 50 epochs.

Bi-Directional LSTM

Next, we make use of recurrent architectures **to capture the temporal relationships** that exist between video frames. Using bidirectional models ensures that information about previous and future frames are captured in each hidden state.

For the "Segment to Label" problem, we train a Bi-Directional LSTM (**model A1**) with max-pooling after the two bi-directional layers. The test accuracy of this model is 49.61%.

For the "Frame to Label" problem, we experiment with both Bi-Directional LSTM and Bi-Directional GRU. The LSTM model (**model B2**) outperforms the GRU model (53% test accuracy compared to 47%). The LSTM model is regularized by using a dropout layer (with *p=0.4*) before the final dense layer and softmax that yields the classified label. It is trained for 25 epochs. Refer to the appendix for the model diagram.

Bi-Directional GRU

A bi-directional GRU (**model A2**) is trained for the "Segment to Label" problem. An LSTM model is trained for the same input, but the GRU model outperforms it (test accuracy of 64.7% compared to 54%). The GRU model uses a hidden vector dimensionality of 160. It is also regularized by using dropout (with *p=0.2*), and is trained for 30 epochs.

For these models, we use the entire video as input. This way, we allow **information from previous segments to be passed to future segments and vice versa**. This retains temporal data that is important for learning the segment labels, as the hidden states **learn what the entire video is about**, and not just about a single segment.

This could be why these models perform better than the LSTM (model A1), which is trained on individual segments instead of the entire video (as individual segments do not contain any information about each other).

Stacked Bi-Directional GRU

Lastly, a stacked bi-directional GRU (**model A3**) is trained for the "Segment to Label" problem. This model takes in the frames of the training videos and passes it through a bi-directional GRU. The feature vectors for the frames in each segment are then pooled using Max- and Average-pooling. These **pooled representations of segments** are then fed into another bi-directional GRU layer, whose hidden states are classified to give labels for each segment of the video.
The second GRU is used to capture better relationships between segments of the video. This is possible as the length of the input sequence is much smaller than the first GRU due to segment wise pooling. The inputs to this second GRU also contain **higher level information** about segments rather than highly granular video frames that are used as inputs to the first GRU.

This model is trained using the Adagrad optimiser and an adaptive learning rate for 25 epochs. The model that produced the lowest Cross Entropy Loss across all epochs is chosen as the best model. It gives a validation accuracy of 77.72% and our best test accuracy of 70.482% on Kaggle.

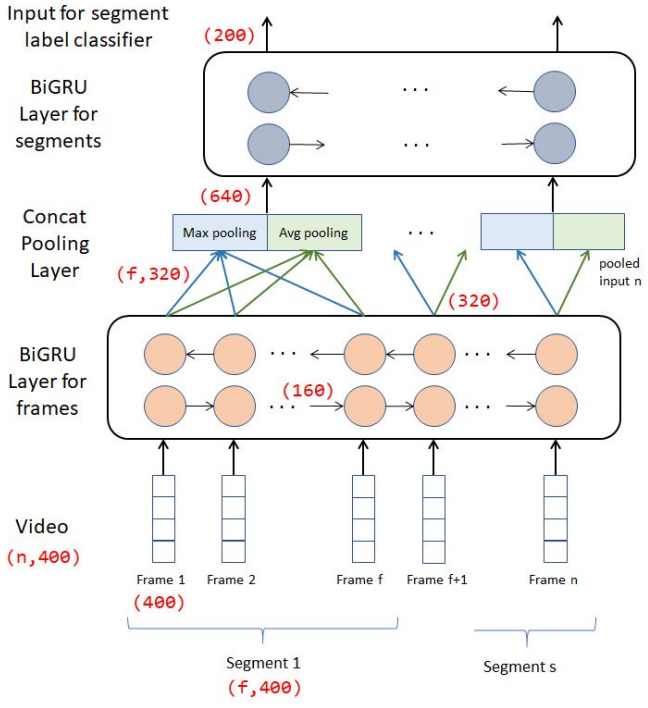

Figure 5: Model A3. Stacked Bi-Directional GRU model. Outputs of the first frame-wise GRU (orange) are pooled and passed through a second segment-wise GRU (gray).

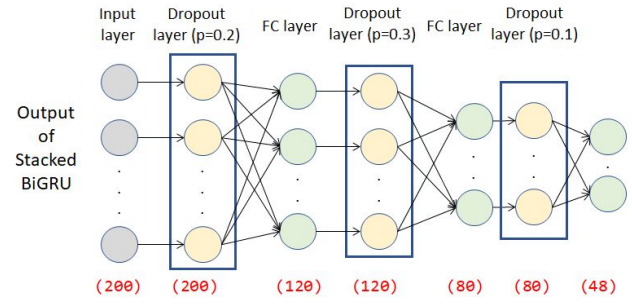

Figure 6: Model A3. Classifier for output from Figure 5.

Label Prediction

The inference task of getting the test labels is straight-forward for the "Segment to Label" task as the final predicted labels correspond to segments.

However, for the "Frame to Label" approach, we must **aggregate the labels of the frames for each segment**. We tried 3 methods - hard voting, soft voting, and a hybrid. For hard voting, we double the count for the middle 70% of the frame labels in a segment when we vote. Soft voting, on the other hand, uses the probabilities that the inference task outputs for the weights.

The final method of voting we try is using the hard weights (as mentioned for hard voting) and the probabilities for each class for a frame in a segment. We multiply the 2 quantities for each such class, for each frame in a segment. The class corresponding to the largest sum is inferred to be the segment's label.

4. Results

	Test Accuracy (Kaggle)	**Input**
Segment to Label		
A1. BiLSTM	49.610%	Segment
A2. BiGRU	64.797%	Entire video
A3. Stacked BiGRU	**70.482%**	Entire video
Frame to Label (Bonus)		
B1. DNN	23.286%	Entire video
B2. BiLSTM	**52.803%**	Entire video

Figure 7: Test accuracies of models

The baseline deep neural network (model B1) gives a test accuracy of 23.286%. Figure 7 plots the model's accuracy and loss across the 50 epochs it is trained for. At the end of every epoch, there is a round of validation.

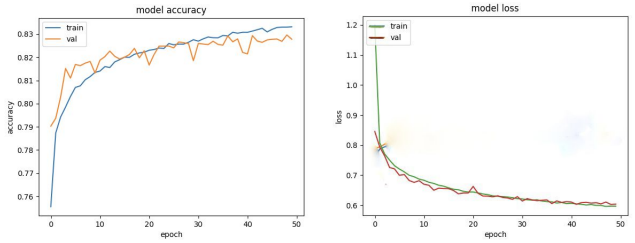

Figure 8: Baseline DNN Model Accuracy and Loss

The Bi-Directional GRU model gives a test accuracy of 64.797%. Figure 8 shows the training and validation loss and accuracy plotted across the 30 epochs it is trained for.

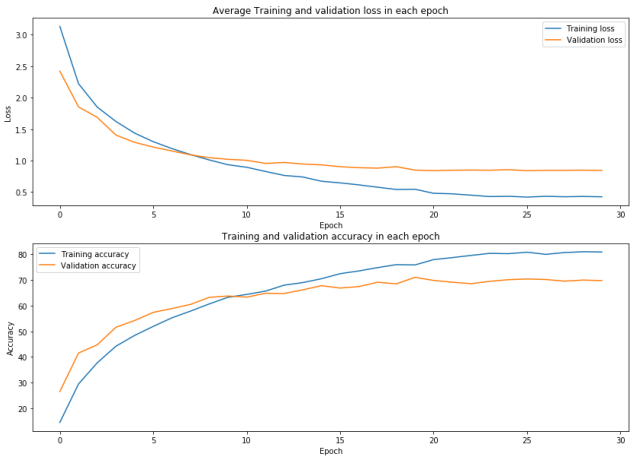

Figure 9: Bi-Directional GRU Model Accuracy and Loss

The **stacked version of the Bi-Directional GRU model is our best model**, giving a test accuracy of 70.482%. Figure 10 shows the training and validation loss and accuracy plotted across 25 epochs. Refer to the Appendix for the confusion matrix produced by testing the model on the validation set.

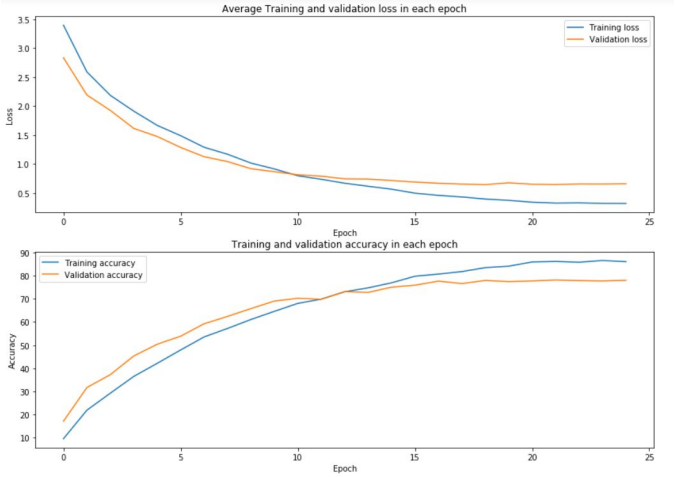

Figure 10: Stacked Bi-Directional GRU Model Accuracy and Loss

5. Discussion

5.1 Recurrent Architecture

The baseline model is a DNN model which achieves a test accuracy of 23% for the "Frame to Label" problem. Fine-tuning the model parameters further is unlikely to improve performance. Take, for instance, increasing the number of neurons in a hidden layer. The resulting model will have more parameters to learn. Unfortunately, the amount of data provided is limited. As there is less data for the training process, it is unlikely that these additional parameters would be optimized sufficiently, leading to stagnant or lower performance.

The nature of our data is sequential - 100s of frames laid out to form a video. These frames carry, to an extent, information about what was present in the previous frame, as well as information about what could be present in the frame to follow. Hence, we proceed with a recurrent neural network architecture for further experiments.

5.2 Bi-Directional Layers

Bi-Directional variants of LSTMs and GRUs provide an important benefit over their vanilla variants - they help understand the context better. By feeding information about the future, the model is given more information to learn from.

5.3 GRU vs LSTM

In Chung, J., Gulcehre, C., Cho, K., & Bengio, Y.'s [4] paper on evaluating gated recurrent networks on sequence modeling, the authors note the differences between the GRU and LSTM cells - specifically that the GRU does not have a mechanism to limit the exposure of the memory content (i.e. the hidden state), while the LSTM can do so with the help of its output gate.

Choosing to use GRUs or LSTMs could have an effect on training. If a GRU cell exposes all of its memory content, the following cells can make use of that information to generate the next hidden state. On the other hand, the LSTM learns how much information is needed to be passed on to the next time-step, and by doing so it is able to prevent an information overload, leading to better performance.

Hence, for each approach we experiment with bi-directional GRUs and bi-directional LSTMs. We note that there is a difference in the performance (5-6% difference) in each approach. However, the specific model trend isn't observed across the two approaches. For the "Frame to Label" approach, the bi-directional LSTM performs better, while for the "Segment to Label" approach, the bi-directional GRU comes out on top. This goes to show that there is **no absolute advantage of using either gated recurrent architecture**, an inference that is inline with Chung et. al's [4] conclusion.

5.4 Pooling

The rationale behind incorporating pooling into our recurrent networks is that it summarises information from features through down-sampling.

The content that we wish to focus on is contained in a few frames, which may occur anywhere in the video segment, or in the video (if we consider the "Frame to Label" problem) . As the segments can consist of hundreds of frames, information may get lost if we only consider the last hidden state of the model. Since we are interested in coming up with a label for a segment, we must take into account information from all frames in the segment, which is why temporal pooling is done. Pooling also converts this **variable-sized feature representation into a fixed-length representation** (which will be used by the classifier).

A max-pool operation is not as susceptible to change when compared to an average pool operation, as the latter depends on all the elements in a portion of the feature representation, while the former only depends on the maximum value. In our "Segment to Label" approach, we experiment with only using a max-pool layer on top of our GRU, as well as using a form of **concat pooling** [3] (concatenation of the max-pooled and average-pooled

outputs) on top of our bi-directional GRU and stacked bi-directional GRU.
In our early experiments with a BiGRU model, average pooling performed slightly better than max pooling with accuracies of 57.90% and 49.61% on the validation and test sets respectively vs 54.8% and 49.45% for max pooling. However, concat pooling showed a significant improvement and had accuracies 65.66% and 56.46% respectively.

5.5 Voting

For hard voting, we observe that the **labels in the middle of each segment provide a better indicator** of what the true label is, which is why we give labels in the middle portion (70% of the segment) a higher weight, and the rest of the 30% (15% on either end of the segment) a lower weight.

Soft voting is implemented with the rationale that we want to use the likelihood of observing a label for a particular frame when it comes to aggregating the label for the segment.

The final method of voting combines the probabilities and the hard weights in an attempt to accentuate the importance of a particular frame's label in a segment.

6. Conclusion

In this paper, we tackled the task of classifying common breakfast actions. We made use of a recurrent neural network architecture and followed two approaches. The "Segment to Label" approach made use of a bi-drectional GRU model that used a concatenation of maximum and average temporal pooling outputs to make the softmax prediction. The harder problem - "Frame to Label", was tackled with a bi-directional LSTM that made use of maximum temporal pooling. We also managed to verify the finding that there is no clear winner when it comes to GRU and LSTM models.

However, there are a few improvements that can be made to our approach. Firstly, using the concept of **self-attention** could make predictions of segments more accurate. Considering each segment as a unit, this would mean that the model would take into account every segment in a video to predict a given segment's label. This is quite intuitive, as it is natural for a segment labeled 'Take Cup' to precede the 'Pour milk' and 'Stir coffee' segments.

Data augmentation is the second proposed task. We were provided with training data from roughly 1500 videos. Even though these gave rise to over 5000 segments, this is still not enough. This could be the reason we faced an issue of stagnating accuracy even on fine tuning the parameters over several attempts.

One way of augmenting our data could be to extract multiple slices of the same segment in a video (for the segment to label approach) or of the same video (for the frame to label approach). This way we would have more training data samples. Furthermore, these slices may or may not overlap with each other. Training our models on such data would make it more robust.

Lastly, we can try **adding the final hidden state** as well to our concat pooling output which only combined max and average pooling outputs. This would add even more useful information for the classifier model as the final hidden state of a sequence represents a summary of the entire sequence.

References

[1] Abu-Mostafa, Yaser M., Magdon-Ismail, Malik and Lin, Hsuan-Tien. (2012) *Learning From Data*, AMLBook.
[2] Bishop, Christopher M. (2006) *Pattern Recognition and Machine Learning*. Springer.
[3] Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification. *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, *1*, 328–339. https://doi.org/10.18653/v1/p18-1031
[4] Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. https://arxiv.org/pdf/1412.3555.pdf

Appendix

1. Model Architecture Diagrams

1.1. Baseline DNN Model

Figure 10 illustrates the architecture of our baseline DNN model. This model aims to solve the 'frame to label' problem.

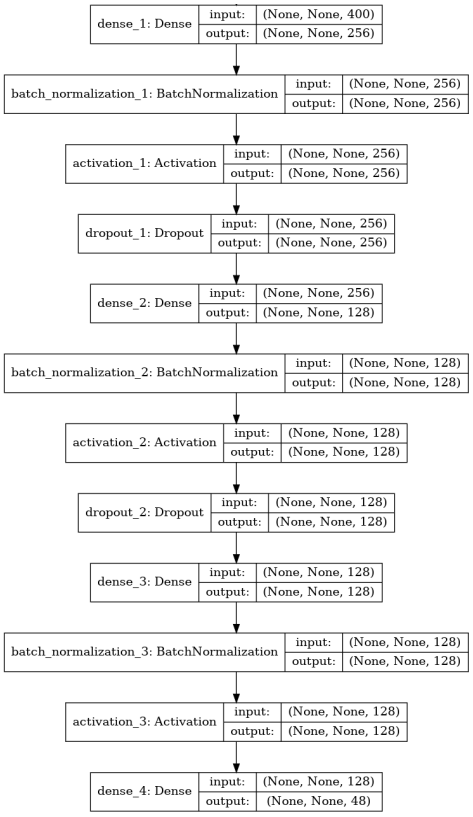

Figure 10: Model B1. Baseline DNN model

1.2. Bi-Directional LSTM Models

Figure 11 illustrates the architecture of the bi-directional LSTM model used for the "Frame to Label" problem.

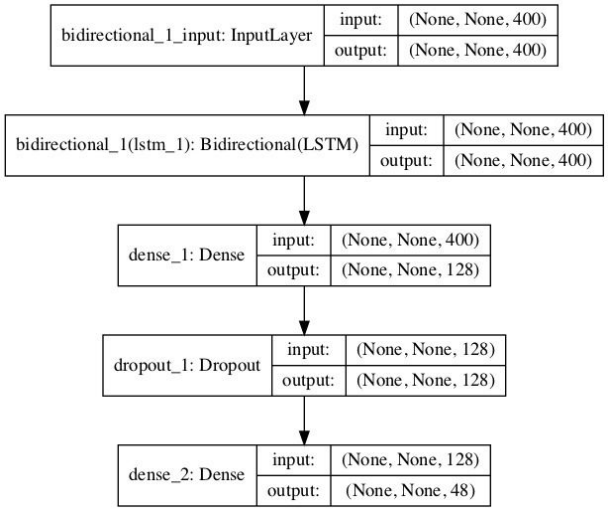

Figure 11: Model B2. Bi-Directional LSTM model

Figure 12 illustrates the architecture of the bi-directional LSTM model used for the "Segment to Label" problem.

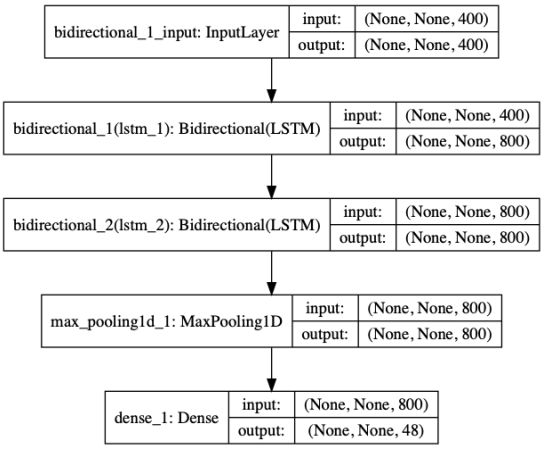

Figure 12: Model A1. Bi-Directional LSTM model

2. Validation Set Results

Figure 13 shows the confusion matrix for the predictions of model A3. Stacked Bi-Directional GRU, our best performing model.

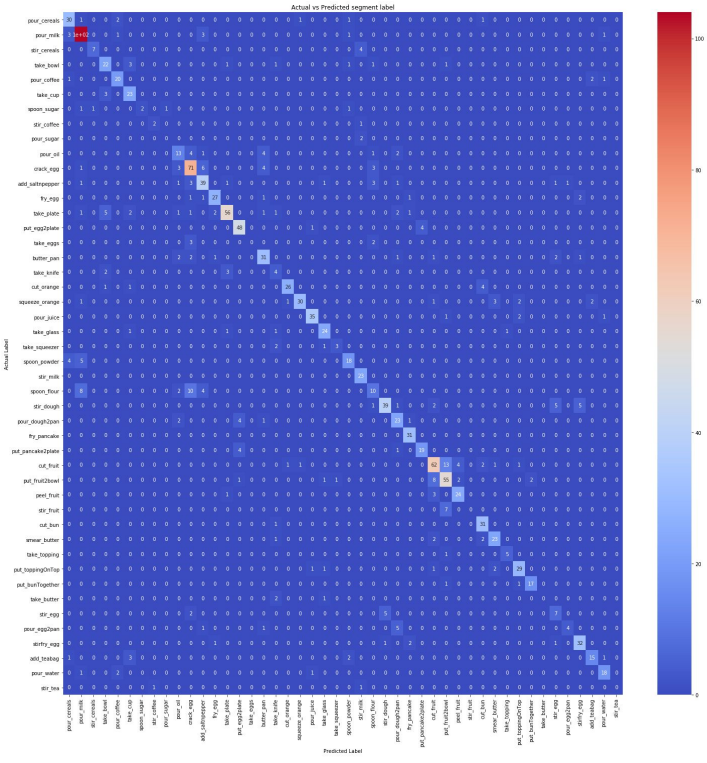

Figure 13: Confusion matrix for Actual labels vs Predicted labels for our best model.