Multi-Label Video Classification with QRNN and DenseNet

To Isaac Zachary

ysto@connect.ust.hk

COMP 4971F (Winter 2018), Supervised by Dr. David Rossiter

1 Introduction

This paper is an extension of the paper "Large-Scale Multi-Label Video Classification Using QRNNs" (To, 2018 [1]). The paper, part of an Undergraduate Independent Studies Project explored using Quasi-Recurrent Neural Networks as a method to predict relevant labels to any given video. A Quasi-Recurrent Neural Network (Bradbury & Merity et al., 2016 [2]) is a type of Neural Network that alternate between Convolutional layers and Pooling layers. This type of Neural Network was chosen due to the architecture being inherently smaller in file size, as well as requiring less computational resources for training.

In this paper, we will introduce a variant of the Quasi-Recurrent Neural Networks introduced in the last paper to predict the relevant labels to any given video. As this problem is a type of sequence classification, meaning that a prediction is made based on a sequence of inputs, we can look into methods that are used in typical sequence classification problems. One of the largest flaws of the previous paper is that the focus was emphasized on the fact that videos are sequences, however, the features of videos and frames were not fully assessed per se, hence a QRNN by itself was not able to achieve a high accuracy.

In this paper, we will combine Densely Connected Convolutional Networks (DenseNet) and QRNNs in an attempt to improve the accuracy of QRNNs. 'Dense Convolution', (Huang, Liu, Maaten, and Weinberger, 2016 [5]) a technique used in Convolutional Networks, introduces skip-connections between QRNN layers. This method is useful in Convolutional Networks as in layers close to the input and the output, shorter connections between said layers are observed to improve accuracy and are more efficient to train.

The experiment presented in this paper is based on the "The 2nd YouTube-8M Video Understanding Challenge" (hereinafter "CHALLENGE") held on Kaggle in 2018 [6], follows the same protocols for evaluation, and compares to the winners of the presented challenge as a benchmark.

The priorities in the selection the model architecture for this task is having one 'light-weight' model with an acceptable accuracy compared to that of other teams on the CHALLENGE. A 'light-weight' model means the model is small in file size and requires minimal computational resources for training and deployment. In the CHALLENGE most teams use multiple models and take a weighted average from the output. This can cause some issues to arise during training and deployment. Hence we would like to keep the model proposed in this paper as simplistic as possible.

2 Technical Methodology

2.1 Dataset

We will use the Youtube-8M dataset [3] for experimentation. The Youtube-8M dataset is the largest publicly available multi-label video classification dataset, with approximately 8 Million videos annotated with 3862 classes of labels [3]. The videos within the dataset averages 3.01 labels per video, where the number of labels per video ranges from 1 to 23. As this dataset covers over 500,000 hours of video, 2.6 billion audio and visual features have been extracted and pre-processed in advance by the Google Research Team as it would be infeasible for research teams to train hundreds of Terabytes worth of video for their model. The next section of this paper will outline the feature extraction process of the Youtube-8M dataset.

2.2 Youtube-8M Feature Extraction

A deep model, namely an Inception network[3], trained on Imagenet was used to pre-process the videos and extract frame-level features. An Inception Network is like a Convolutional Neural Network except it has been heavily modified to boost results and performance. The following points are the procedures and specifications of the feature extraction process of this dataset:

- Each video was decoded at 1 frame-per-second up to the first 360 seconds (6 minutes). This cap was implemented for storage and computation reasons
- Decoded frames are fed into the Inception Network
- The Rectified Linear Unit (ReLU) activation function of the last hidden layer before the classification layer is fetched
- The feature vector is 2048-dimensional per second of video
- Principal Component Analysis (PCA) along with whitening was applied to scale down the feature dimensions to 1024, followed by quantization (1 byte per coefficient)
- PCA and Quantization downsamples the size of the data by the factor of 8
- The mean vector and covariancce matrix for PCA was computed on all frames from the training partition
- Each 32-bit float was quantized into 256 distinct values (8 bits) using optimally computed (non-uniform) quantization bin boundaries

- The dataset is then broken into 3844 shards each for the training, validation, and testing partitions
- The dataset is separated into the Frame-level features dataset, which is approximately 1.53 Terabytes, and the Video-level features dataset, which is approximately 31 Gigabytes

The features extracted were also split into two types and two different datasets. Namely frame-level features and video-level representations. For frame-level features, 20 random frames are randomly sampled from each video, then the ground truth labels are associated with each frame. Video-level representations are simply fixed-length video features being extracted using the above procedures.

2.3 Evaluation

In order to prevent bias in the evaluation process, the evaluation protocol [3] provided by Google will be used to calculate the accuracy of a given model. (Note: It was simplified to enable faster calculations of important metrics) The three protocols that are used to evaluate the accuracy of the model are Global Average Precision (GAP), Hit @ k, where k = 1, and Precision at equal recall rate (PERR).

2.3.1 Global Average Precision (GAP)

Global Average Precision (GAP) at k, where k = 20 is the primary evaluation metric [3]. A list of the top 20 predicted labels for each video and their corresponding confidence scores are generated after testing. The evaluation treats each predicted label and the confidence score as an individual data point to compute the GAP across all the predictions and videos. If there are N predictions sorted by decreasing confidence scores, the Global Average Precision is computed by

$$\text{GAP} = \sum_{i=1}^{N} p(i) \triangle r(i)$$

where N is the number of final predictions made, p(i) is the precision and r(i) is the recall.

2.3.2 Hit @ k

This is the percentage of test samples that contain at least one correct label in the top k predictions made by the model. In this case we will be using k = 1 to evaluate the accuracy of the model. Where $\text{rank}_{v,e}$ is the rank of entity e on video v, where the label with the highest confidence score is rank 1. G_v is the set of ground-truth labels for v, Hit@K can be represented as:

$$\frac{1}{|V|} \sum_{v \in V} \vee_{e \in G_v} \mathbb{I}(\text{rank}_{v,e} \leq k),$$

where $\vee$ is logical OR

2.3.3 Precision at equal recall rate (PERR)

For Precision at equal recall rate (PERR), we retrieve the same number of labels per video as there are in the ground-truth to measure the video-level annotation precision. With the same notation as Hit@k, PERR can be represented as:

$$\frac{1}{|V : |G_v| > 0|} \sum_{v \in V : |G_v| > 0} \left[\frac{1}{|G_v|} \sum_{e \in G_v} \mathbb{I}(\text{rank}_{v,e} \leq |G_v|) \right].$$

3 Model Architecture

3.1 Architecture of QRNN

In the previous work [1], a Quasi-Recurrent Neural Network (QRNN) [2] was chosen to tackle the task at hand. QRNNs alternates between convolution layers and pooling layers. Where the convolutional layer computes and maps current representations, and the pooling layers are used to handle sequential dependencies. This architecture is represented in Figure 1.

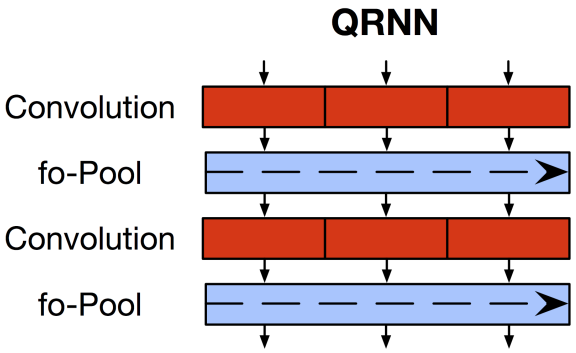

Figure 1: The QRNN Architecture

3.2 Architecture of DenseNet

In typical Convolutional Neural Networks (CNNs), the output of the n^{th} layer is connected to the $(n + 1)^{th}$ layer as its input. This is represented in Figure 2.

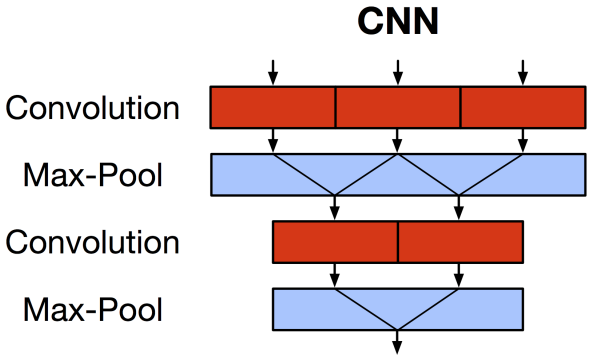

Figure 2: Typical CNN Architecture

For DenseNet [5], instead of each layer only having a convolutional connection to one other layer, each layer is connected to all its following layers. In terms of n, instead of the output of the nth layer being only connected to the input of the (n + 1)th layer, the nth layer receives feature-maps from all its preceding layers, $x_0, ...x_{n-1}$ as input. We refer to this architecture as Dense Blocks. This is denoted as follows:

$$\mathrm{x}_n = H_n([\mathbf{x_0}, \mathbf{x_1}, ..., \mathbf{x_{n-1}}])$$

where
x denotes a single fixed vector input
$\mathbf{x_n}$ denotes the output of the nth layer
H_n denotes a composite function of operations

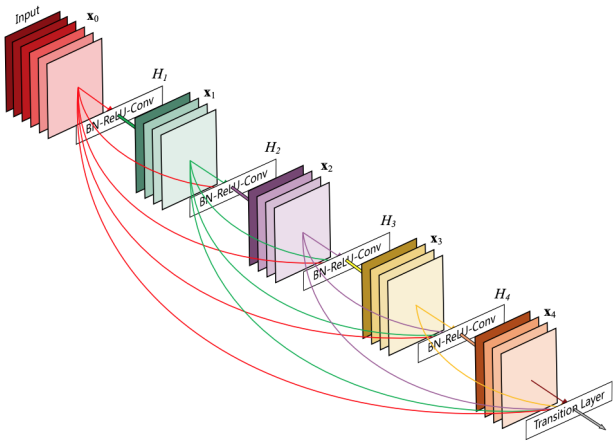

Figure 3: Architecture of DenseNet [5]

H_n is defined as a composite function consisting of three consecutive operations: batch normalization, rectified linear unit (ReLU), and a convolution, in that order. The inputs of H_n is concatenated into a single tensor.

In any case where the concatenation operation in the above equation is not viable, down-sampling layers by changing the size of feature maps. In order to facilitate this operation, Dense Blocks are used. Between Dense Blocks we insert transition layers. Typically these layers consists of a Batch Normalization layer, a ReLU layer, a convolutional layer, a pooling layer, and is then regularized with a technique called dropout. Dropout omits hidden and visible units (neurons) in a neural network during training. This technique is used to prevent overfitting. In the last layer we insert a sigmoid function, this is for us to generate the confidence score for each prediction made by the model.

In the function H_n, it produces k feature-maps. The number of input feature-maps that the nth layer has is denoted by:

$$\mathrm{k}_0 + k * (n - 1)$$

where
k_0 denotes the number of channels in the input layer

The hyperparameter k regulates the amount of new information that is contributed to the global state by each layer. The global state of the network is the feature maps present within the DenseNet. One of the strengths of DenseNet is that the global state can be accessed by any layer of the network, instead of having to replicate it layer by layer like traditional neural networks.

3.3 Densely-Connected QRNN

The next step was to implement the dense convolution techniques to the QRNN architecture. It was suggested in Bradbury & Merity et al., 2016 [2] that connecting each QRNN layer's output to the following layers as outputs, is the equivalent to including connections between the input embeddings, every QRNN layer and every pair of QRNN layers. This is true as the global state of the network is still present within the QRNN and accessible by any layer of the network.

The architecture of the Densely-Connected QRNN (dcQRNN) is similar to that of DenseNet. dcQRNN blocks were created and connected, and between the blocks, transition layers were inserted. However, the transition layers of dcQRNN only consisted of a Batch Normalization layer, a ReLU layer, and a dropout "layer". A sigmoid function was inserted at the output of the dcQRNN as well.

4 Experiments

4.1 Training

The models were trained on a Macbook Pro with a 2.8 GHz Intel Core i5 CPU and 8 GB 1600 MHz DDR3 memory with all shards of the Youtube-8M dataset. The models were built on Keras, an open-sourced Python Neural Network library on top of Tensorflow. Tensorflow, another open-sourced framework was used to train and test the models. The models were exclusively trained on the CPU of the device. All models were trained on 5 epochs to shorten the training time required. All models were built using the console Sublime Text 3.1.1 and trained on the Terminal provided by macOS High Sierra Version 10.13.

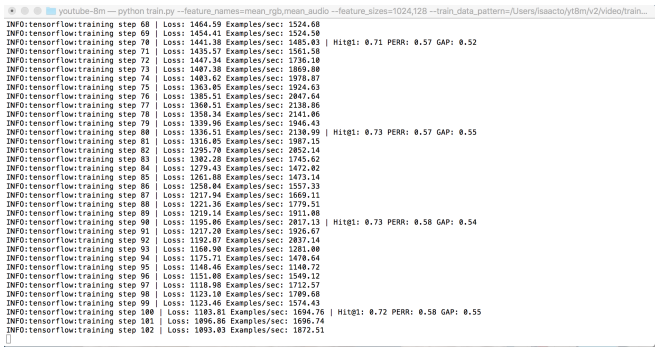

Figure 4: An illustration of the training in progress

4.2 Results

As aforementioned to prevent bias in the evaluation process, Google's evaluation protocol was used to evaluate the accuracy of the model.

The results yielded:

Level	Model	GAP
Video	DenseNet (This Study)	86.7
Video	dcQRNN (This Study)	85.1
Video	QRNN (Previous Study)	80.8

4.3 Comparison of results

The results from the models was compared to the winners of the Google Cloud YouTube-8M Video Understanding Challenge 2017 as well as the winners of the 2nd YouTube-8M Video Understanding Challenge. They include teams from Samsung, Baidu and Tsinghua University.

Year	Rank	Model	GAP
2018	1	Ensemble of 13 models [15]	88.987
2018	2	Ensemble of 16 models [14]	88.729
2018	3	Ensemble of 3 models [13]	88.722
2019	n/a	DenseNet (This Study)	86.7
2019	n/a	dcQRNN (This Study)	85.1
2017	1	Ensemble of 25 models [9]	84.966
2017	2	Weighted 74 models [10]	84.589
2017	3	Ensemble of of 57 models [11]	84.541
2018	n/a	QRNN (Previous Study) [1]	80.8

Where ”Rank” represents the ranking that model gained in the challenge.

As evident from the Global Average Prediction scores, the QRNN falls shy against the results achieved by the winners from 2017. This may be because the winners selected to use multiple models instead of one model, whereas an ensemble method can be explored in the future. The DenseNet performs quite close to the winners from 2018, while the dcQRNN exceeds the winners from 2017 but yet it performs rather unremarkably when compared to the winners from 2018.

5 Implications

Observing the results obtained from this study and comparing it against the previous study [1], it is evident that the approach of using QRNNs may be inferior to other approaches, specifically DenseNets. While the QRNN and dcQRNN performed at an average performance, the DenseNet outperformed both models. This could be because in a task such as making predictions based on certain videos, the emphasis on feature mapping from fixed vector inputs is higher than the sequential aspect of it.

From the results, it is also evident that introducing connections between the input embeddings, every QRNN layer and every pair of QRNN layers, and thus writing a global state accessible by all of the layers increases the accuracy of the predictions made.

6 Further Exploration

6.1 Expanding upon the model

In the future I would like to test how the results of an ensemble of models would differ from the result from one model, as evidently one single model may not be able to compete with the results of an ensemble of models. However, an issue to deal with when working with an ensemble of models is the weighting of certain models.

Another way to potentially increase accuracy is to pre-process the data and to ’clean’ the noise present within the dataset. Certain techniques such as training the models on soft labels could be helpful to increasing the accuracy of the models. There are also multiple regularization techniques applicable to neural networks, it would be helpful to experiment with different types of regularization techniques and their effect on the overall accuracy of the models. Last but not least, the frame-level dataset has not been explored in this study yet. Whether both datasets can complement each other is a question worth exploring in the future.

References

[1] Isaac Zachary To. Large-Scale Multi-Label Video Classification Using QRNNs. 2018. Re-trieved from http://www.cse.ust.hk/ ∼ rossiter/independent_studies_projects/video_classification_qrnn/video_classification_qrnn_report.pdf

[2] James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. Quasi-recurrent neural networks. CoRR, abs/1611.01576, 2016.

[3] Sami Abu-El-Haija, Nisarg Kothari, Joonseok Lee, Paul Natsev, George Toderici, Balakrishnan Varadarajan, and Sudheendra Vijayanarasimhan. Youtube-8m:A large-scale video classification benchmark. CoRR, abs/1609.08675, 2016.

[4] YouTube-8M: A Large and Diverse Labeled Video Dataset for Video Understanding Research. (n.d.). Retrieved from https://research.google.com/youtube8m/

[5] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely Connected Convolutional Networks. CoRR, abs/1608.06993, 2016.

[6] The 2nd YouTube-8M Video Understanding Challenge. (n.d.). Retrieved from https://www.kaggle.com/c/youtube8m-2018/

[7] Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 5(2), 157–166.

[8] Google Cloud YouTube-8M Video Understanding Challenge. (n.d.). Retrieved from https://www.kaggle.com/c/youtube8mevaluation

[9] Antoine Miech, Ivan Laptev, and Josef Sivic. Learnable pooling with context gating for video classification. CoRR, abs/1706.06905, 2017.

[10] He-Da Wang, Teng Zhang, and Ji Wu. The monkeytyping solution to the youtube-8m video understanding challenge. CoRR, abs/1706.05150, 2017.

[11] Fu Li, Chuang Gan, Xiao Liu, Yunlong Bian, Xiang Long, Yandong Li, Zhichao Li, Jie Zhou, and Shilei Wen. Temporal modeling approaches for large-scale youtube-8m video understanding. CoRR, abs/1707.04555, 2017.

[12] M. I. Jordan. Hierarchical mixtures of experts and the em algorithm. Neural Computation, 6, 1994.

[13] Rongcheng Lin, Jing Xiao, and Jianping Fan. NeXtVLAD: An Efficient Neural Network to Aggregate Frame-level Features for Large-scale Video Classification. CoRR, abs/1811.05014, 2018.

[14] Pavel Ostyakov, Elizaveta Logacheva, Roman Suvorov, Vladimir Aliev, Gleb Sterkin, Oleg Khomenko, and Sergey I. Nikolenko. Label Denoising with Large Ensembles of Heterogeneous Neural Networks. CoRR, abs/1809.04403, 2018.

[15] Skalic, M., Austin, D. (2019). Building A Size Constrained Predictive Models for Video Classification. Lecture Notes in Computer Science Computer Vision – ECCV 2018 Workshops, 297-305. doi:10.1007/978-3-030-11018-5_27