AN ATTEMPT AT LARGE SCALE MULTI-LABEL VIDEO CLASSIFICATION USING THE QRNN ARCHITECTURE

Ismael Goulani
Isgoulani@gmail.com

The Hong Kong University of Science and Technology

COMP 4971C (Fall 2018), Supervised by Dr. David Rossiter

Abstract. In this study, we present an attempt of using the QRNN architecture for video classification tasks. Previous usage of QRNN was on language modeling tasks such as sentiment analysis, next-word prediction and translation. With its massive use of parallel computation, it's said to be up to 16 times faster than the old approach Bradbury & Merity et al, 2016 [1]. Our goal is to show through this study that QRNNs can also be used for video classification and can produce good performances as other state of the art Network architecture like LSTMs and GRUs. We first start by making an overview of video classification tasks, the YouTube-8M video datasets and the Kaggle challenge held in 2018. We will also present various architecture used for this tasks and also explain the QRNN architecture.

Keywords: Deep learning, Multi-label classification, QRNN, Video processing

1. Introduction

From social networks like Facebook, YouTube or snapchat to traffic surveillance and cameras security systems, video generation is on the rise.
Paired with the recent advances in Deep learning, specifically on video and audio processing, it opens the way for a better understanding of videos contents.
However, the designing of the best performing Deep learning algorithms are most of the time computationally expensive, in terms of speed and memory usage. Hence, putting these models in production can require a lot of Graphic Process units(GPUs).
In the 2nd edition of the "Google Cloud & YouTube-8M Video Understanding Challenge" held on Kaggle in 2018, participants were asked to come up with a sized constraints model that fits within a 1GB file.

1.1. Data

Our experiment will lay on the largest multi-label video classification dataset publicly available provided by Google, the YouTube-8M dataset [2].
This iteration of the YouTube-8M dataset contains 6.1 million samples which represents over 350000 hours of videos split into 3 partitions as follow :

- 70% for the training set
- 20% for the validation set
- 10% for the test set

The videos in the dataset are labeled with 3862 tags with an average of 3 tags per video. Each video includes up to 360 frames. Each frame consist of 1024 and 128 visual and audio features.
The visual features were extracted from the last ReLU activated layer prior to classification in the Inception-v3 Network trained on ImageNet. The audio features were extracted using a VGG inspired acoustic model described in Hershey et al [3].
Dimensionality reduction and quantization were applied on Both visual and audio features to reduce the storage coast.
The total size of the combined features is approximatively 2TB.

1.2.Evaluation

To avoid any bias regarding the evaluation process, we decided to follow the evaluation methods provided by Google during the 2018 Kaggle competition.
The main evaluation metric for this competition is the Global Average Precision(GAP).

$$GAP = \sum_{i=0}^{N} p(i)\Delta r(i)$$

Where N is the number of final predictions, $p(i)$ is the precision, and $r(i)$ is the recall.
In the evaluation, $N = 20 * (Number\ of\ videos)$ were used.

In addition to the GAP, some additional metrics where also used:
- Mean Average Precision (MAP) :

Knowing that the examples are not uniformly distributed over labels in the dataset, MAP is used instead of AUC. MAP computes mean-per-class AUC of precision-recall curves.

Formally, we have :

$$MAP = \frac{1}{E}\sum_{e\epsilon E} APe$$

- Hit@k :

It's the fraction of test samples that contains at least one of the ground truth labels in the top k predictions. For this experiment, we use k=1 to evaluate the accuracy of the model.

$$Hit@k = \frac{1}{|V|}\sum_{v\in V} \vee\ e \in Gv\ \mathbb{I}(rankv, e \leq k)$$

- Precision at Equal Recall Rate(PERR):

It's similar to MAP but instead of using a fixed k=20, we compute the mean precision up to the number of ground truth labels in each class.

$$PERR = \frac{1}{|V|}\sum_{v\in V}[\frac{1}{|Gv|}\sum_{k\in Gv} \mathbb{I}(rankv, k \leq |Gv|]) \text{ ,}$$

Where Gv is the set of ground truth labels for video v and $\mathbb{I}(rankv, k \leq |Gv|)$ counts the number of correct predictions made within the top $|Gv|$.

2. Model architecture

2.1.Background

Before diving into the Architecture, we will first talk about the Classique architecture used for the classification tasks like Neural Networks, CNNs and LSTMs.

To put it simply, a neural network is a superposition of layers of cells called "neurons" consisting of an input layer and an output layer.
The lower layers are interconnected to the upper layers through weighted connections, i.e. each connection of the network has a weight.
The values of the lower layer are multiplied by the different weights of the interconnections, then transmitted to the next layer to which, each neuron of the layer adds a value called "bias" and takes a "decision" by applying a function called "activation function" . The results thus obtained at the end of the decisions will be transmitted to the next layer. The outputs of the lower layer become the inputs of the upper layer.
The same process is repeated until the output layer makes the final decision.

Convolutional neural networks CNNs [4] are quietly similar to neural networks.
Indeed all the underlying concepts of neural networks like activation functions, loss function forward/backward propagation etc, also applies to CNNs.
The difference is that CNNS make the assumption that the input is a fixed sized vector like an image.

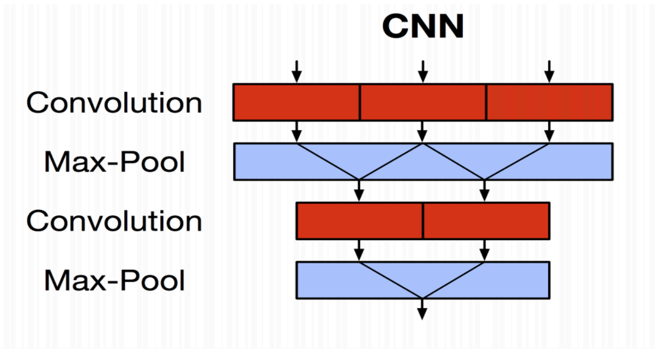


Figure 1: CNN architecture

Typically, CNN is composed of blocs:

- A convolution block called CONV composed of filters :

The filters are matrices of size smaller than the matrices of the pixels of the input image. They make it possible to determine particular shapes on the image by sliding over the entire surface of the image.
Indeed, the shapes are detected after performing some convolutions calculations between the filters and the matrix of pixel of the image. If the resulting values of the convolution calculations are high, then it means that the shapes represented by the filters are present on the image.

- The so-called "Pooling” layer Max-Pool :

It is responsible for reducing the volume of input data for reasons of speed of learning, both in height and width.

- The fully connected layer :

Coming after the last Max-Pool layer, this layer is the same as that used in a conventional neural network. Before entering this layer, the data is concatenated into a 1D size matrix. It allows to make the classification. Despite CNNs are great for handling task like image classification, object detection et cetera, CNNs are not suits for handling sequential data where a fourth dimension has to be take into account , the time dimension. Hence, Recurrent neural networks RNNs Andrej Karpathy, Justin Johnson, and Fei-Fei Li (2015) [5] were introduced to handle these kind of data.
RNNs operates over a sequence of vector in the input, the output or both. But the problem is, RNNs suffer from Long-term dependencies .
Since the RNNs Network tends to be deep if we consider one timestep as a layer, when performing backpropagation, the weights at the beginning of the network tends to change slowly and the network stop to learn at these layers: This is the “Vanishing Gradient” problem Bengio et al. (1994)[6]. RNNs only looks at recent information to perform present task. To handle this issue, Long Short Term Memory (LSTMs) Hochreiter & Schmidhuber (1997) [7] are introduced.

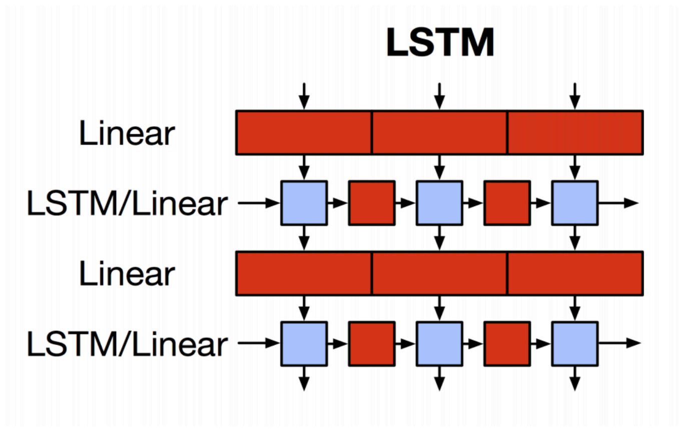


Figure 2 : LSTM architecture

By maintaining a memory vector via a gating mechanism, LSTM allows past information to be read, written or reset after a long period of time. Useful information are kept in memory and that help handle the vanishing gradient problem.

2.2. Architecture

As the title of this study suggest, we choose to use the QRNN architecture to handle this classification task.

QRNNs are built using three components used in Neural Networks:

- A convolution neural network:

Used to compute intermediate vectors and gating vectors.

- A pooling layer :

It handles sequential dependencies. Applied on the inputs computed by the CNN layer, this layer computes the hidden states(outputs). The architecture is represented in the figure below.

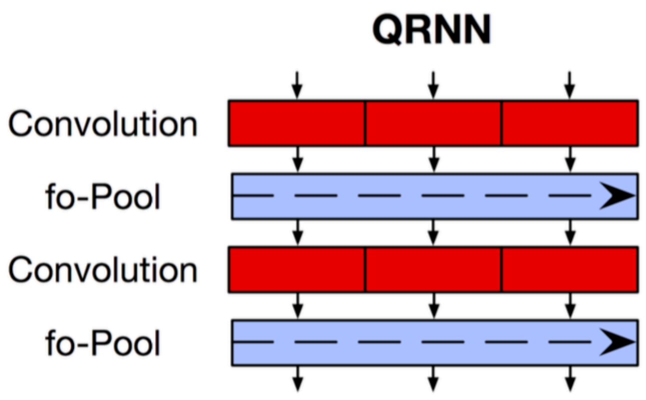


Figure 3 : QRNN architecture

The combination of the strengths of basic CNNs and LSTMs, made this architecture a good choice to handle this video classification challenge.

Indeed, since each video is a sequence of frame, it's an evidence that this sequence should be handle by sequential models like RNNs and LSTMs, this role is played by the pooling layer in the QRNN architecture.
In addition to that, we know that the frames are essentially images and CNN is the de-facto choice to handle image data. Which plays the strengths of CNN layers in the architecture.

2.2.1 Convolutional layer

This layer is used by QRNN in the timestep dimension to compute three vectors : a candidate vector, a forget gate, and an output gate.

When an input sequence of n-dimensional vectors $x_1, x_2, \ldots x_T$ is given, the convolutional layer for the candidate vectors with m filters produces a sequence of T m-dimensional output vectors $z_1, z_2, \ldots z_T$. The forget gates and the output gates follows the same process. The three vectors can be represented by these equations:

$$z_t = \tanh(conv_{w_z}(x_t, \ldots, x_{t-k+1}))$$

$$f_t = \sigma(conv_{w_f}(x_t, \ldots, x_{t-k+1}))$$

$$z_t = \sigma(conv_{w_o}(x_t, \ldots, x_{t-k+1}))$$

Where *conv* represents convolution and k is the filter size.

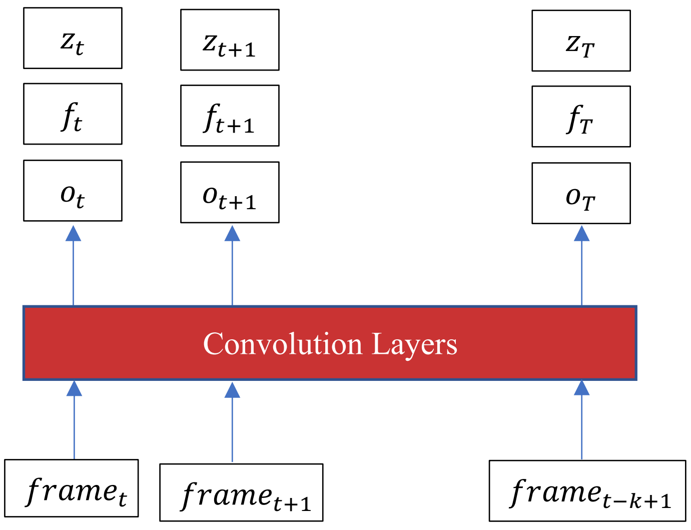


Figure 4 : Applying convolutions to compute intermediate vectors and gating vectors

2.2.2 Pooling layer

Using the output from the convolutional layers, the pooling layers compute the hidden states(outputs) as follow :

$$c_t = f_t \odot c_{t-1} + (1 - f_t) \odot z_t$$

$$h_t = o_t \odot c_t$$

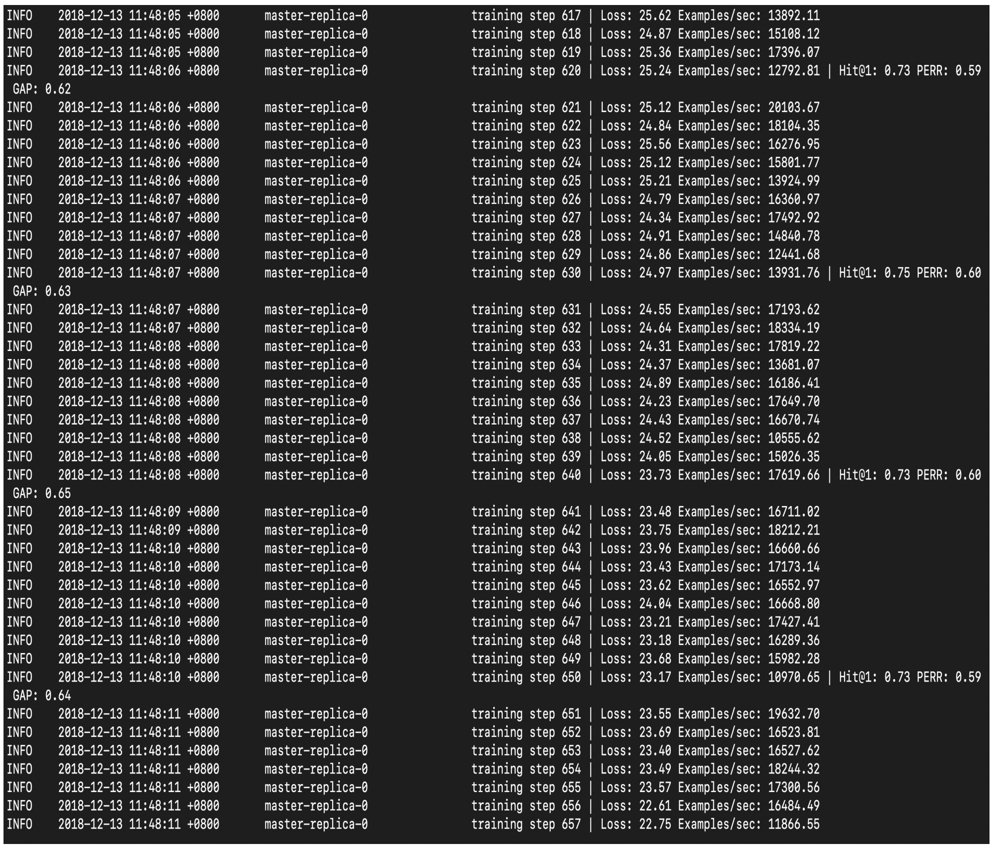
Where $\odot$ represents element-wise multiplication.

The pooling layer equations seems to be similar as the equations of a LSTM unit, and may cause some confusion of the role of the role of the pooling layers in QRNNs.
The main difference between them resides in the fact that all the sequential processing occurs only in the pooling layers. The values computed by the convolution layers z_t, f_t and o_t do not depend of previous values in the network.
This is actually where the magic of QRNN occur. All the heavy computation is done in parallel and the sequential processing is handle by only on component in the network : the pooling layers.

3. Experiments

3.1. Training details

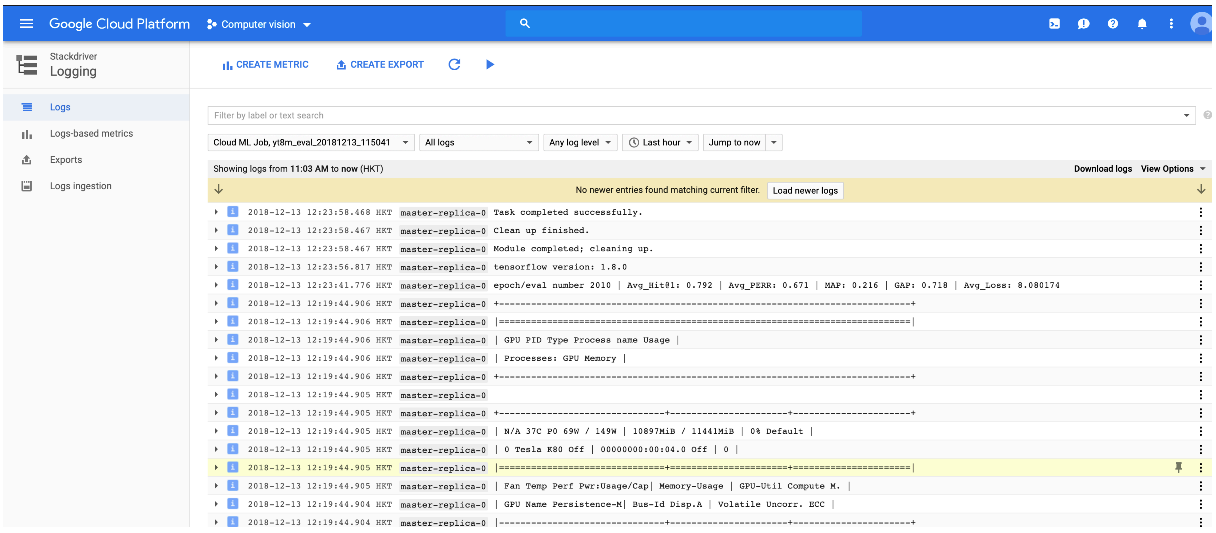
The model was trained on Google Cloud with an ubuntu virtual machine.
The virtual machine had 1 Tesla K80 GPU, 26 GB of ram memory. The model was built, train and test using all the shrads of the Youtube-8M dataset with Tensorflow an open source deep learning framework developed by Google and the python programming language. The training job was sent to Google cloud using the terminal of a Macbook Pro with 2.6 GHz intel core i7 CPU and 16GB 2133 MHz DDR3.



```
INFO    2018-12-13 11:48:05 +0800       master-replica-0            training step 617 | Loss: 25.62 Examples/sec: 13892.11
INFO    2018-12-13 11:48:05 +0800       master-replica-0            training step 618 | Loss: 24.87 Examples/sec: 15108.12
INFO    2018-12-13 11:48:05 +0800       master-replica-0            training step 619 | Loss: 25.36 Examples/sec: 17396.07
INFO    2018-12-13 11:48:06 +0800       master-replica-0            training step 620 | Loss: 25.24 Examples/sec: 12792.81 | Hit@1: 0.73 PERR: 0.59
 GAP: 0.62
INFO    2018-12-13 11:48:06 +0800       master-replica-0            training step 621 | Loss: 25.12 Examples/sec: 20103.67
INFO    2018-12-13 11:48:06 +0800       master-replica-0            training step 622 | Loss: 24.84 Examples/sec: 18104.35
INFO    2018-12-13 11:48:06 +0800       master-replica-0            training step 623 | Loss: 25.56 Examples/sec: 16276.95
INFO    2018-12-13 11:48:06 +0800       master-replica-0            training step 624 | Loss: 25.12 Examples/sec: 15801.77
INFO    2018-12-13 11:48:06 +0800       master-replica-0            training step 625 | Loss: 25.21 Examples/sec: 13924.99
INFO    2018-12-13 11:48:07 +0800       master-replica-0            training step 626 | Loss: 24.79 Examples/sec: 16360.97
INFO    2018-12-13 11:48:07 +0800       master-replica-0            training step 627 | Loss: 24.34 Examples/sec: 17492.92
INFO    2018-12-13 11:48:07 +0800       master-replica-0            training step 628 | Loss: 24.91 Examples/sec: 14840.78
INFO    2018-12-13 11:48:07 +0800       master-replica-0            training step 629 | Loss: 24.86 Examples/sec: 12441.68
INFO    2018-12-13 11:48:07 +0800       master-replica-0            training step 630 | Loss: 24.97 Examples/sec: 13931.76 | Hit@1: 0.75 PERR: 0.60
 GAP: 0.63
INFO    2018-12-13 11:48:07 +0800       master-replica-0            training step 631 | Loss: 24.55 Examples/sec: 17193.62
INFO    2018-12-13 11:48:07 +0800       master-replica-0            training step 632 | Loss: 24.64 Examples/sec: 18334.19
INFO    2018-12-13 11:48:08 +0800       master-replica-0            training step 633 | Loss: 24.31 Examples/sec: 17819.22
INFO    2018-12-13 11:48:08 +0800       master-replica-0            training step 634 | Loss: 24.37 Examples/sec: 13681.07
INFO    2018-12-13 11:48:08 +0800       master-replica-0            training step 635 | Loss: 24.89 Examples/sec: 16186.41
INFO    2018-12-13 11:48:08 +0800       master-replica-0            training step 636 | Loss: 24.23 Examples/sec: 17649.70
INFO    2018-12-13 11:48:08 +0800       master-replica-0            training step 637 | Loss: 24.43 Examples/sec: 16670.74
INFO    2018-12-13 11:48:08 +0800       master-replica-0            training step 638 | Loss: 24.52 Examples/sec: 10555.62
INFO    2018-12-13 11:48:08 +0800       master-replica-0            training step 639 | Loss: 24.05 Examples/sec: 15026.35
INFO    2018-12-13 11:48:08 +0800       master-replica-0            training step 640 | Loss: 23.73 Examples/sec: 17619.66 | Hit@1: 0.73 PERR: 0.60
 GAP: 0.65
INFO    2018-12-13 11:48:09 +0800       master-replica-0            training step 641 | Loss: 23.48 Examples/sec: 16711.02
INFO    2018-12-13 11:48:09 +0800       master-replica-0            training step 642 | Loss: 23.75 Examples/sec: 18212.21
INFO    2018-12-13 11:48:10 +0800       master-replica-0            training step 643 | Loss: 23.96 Examples/sec: 16660.66
INFO    2018-12-13 11:48:10 +0800       master-replica-0            training step 644 | Loss: 23.43 Examples/sec: 17173.14
INFO    2018-12-13 11:48:10 +0800       master-replica-0            training step 645 | Loss: 23.62 Examples/sec: 16552.97
INFO    2018-12-13 11:48:10 +0800       master-replica-0            training step 646 | Loss: 24.04 Examples/sec: 16668.80
INFO    2018-12-13 11:48:10 +0800       master-replica-0            training step 647 | Loss: 23.21 Examples/sec: 17427.41
INFO    2018-12-13 11:48:10 +0800       master-replica-0            training step 648 | Loss: 23.18 Examples/sec: 16289.36
INFO    2018-12-13 11:48:10 +0800       master-replica-0            training step 649 | Loss: 23.68 Examples/sec: 15982.28
INFO    2018-12-13 11:48:10 +0800       master-replica-0            training step 650 | Loss: 23.17 Examples/sec: 10970.65 | Hit@1: 0.73 PERR: 0.59
 GAP: 0.64
INFO    2018-12-13 11:48:11 +0800       master-replica-0            training step 651 | Loss: 23.55 Examples/sec: 19632.70
INFO    2018-12-13 11:48:11 +0800       master-replica-0            training step 652 | Loss: 23.69 Examples/sec: 16523.81
INFO    2018-12-13 11:48:11 +0800       master-replica-0            training step 653 | Loss: 23.40 Examples/sec: 16527.62
INFO    2018-12-13 11:48:11 +0800       master-replica-0            training step 654 | Loss: 23.49 Examples/sec: 18244.32
INFO    2018-12-13 11:48:11 +0800       master-replica-0            training step 655 | Loss: 23.57 Examples/sec: 17300.56
INFO    2018-12-13 11:48:11 +0800       master-replica-0            training step 656 | Loss: 22.61 Examples/sec: 16484.49
INFO    2018-12-13 11:48:11 +0800       master-replica-0            training step 657 | Loss: 22.75 Examples/sec: 11866.55
```

Figure 5 : An illustration of the training process

3.2.Results



Showing logs from 11:03 AM to now (HKT)

No newer entries found matching current filter.

```
2018-12-13 12:23:58.468 HKT master-replica-0 Task completed successfully.
2018-12-13 12:23:58.467 HKT master-replica-0 Clean up finished.
2018-12-13 12:23:58.467 HKT master-replica-0 Module completed; cleaning up.
2018-12-13 12:23:56.817 HKT master-replica-0 tensorflow version: 1.8.0
2018-12-13 12:23:41.776 HKT master-replica-0 epoch/eval number 2010 | Avg_Hit@1: 0.792 | Avg_PERR: 0.671 | MAP: 0.216 | GAP: 0.718 | Avg_Loss: 8.080174
2018-12-13 12:19:44.906 HKT master-replica-0 +-----------------------------------------------------------------------------+
2018-12-13 12:19:44.906 HKT master-replica-0 |=============================================================================|
2018-12-13 12:19:44.906 HKT master-replica-0 | GPU PID Type Process name Usage |
2018-12-13 12:19:44.906 HKT master-replica-0 | Processes: GPU Memory |
2018-12-13 12:19:44.906 HKT master-replica-0 +-----------------------------------------------------------------------------+
2018-12-13 12:19:44.905 HKT master-replica-0
2018-12-13 12:19:44.905 HKT master-replica-0 +-------------------------------+----------------------+----------------------+
2018-12-13 12:19:44.905 HKT master-replica-0 | N/A 37C P0 69W / 149W | 10897MiB / 11441MiB | 0% Default |
2018-12-13 12:19:44.905 HKT master-replica-0 | 0 Tesla K80 Off | 00000000:00:04.0 Off | 0 |
2018-12-13 12:19:44.905 HKT master-replica-0 |===============================+======================+======================|
2018-12-13 12:19:44.905 HKT master-replica-0 | Fan Temp Perf Pwr:Usage/Cap| Memory-Usage | GPU-Util Compute M. |
2018-12-13 12:19:44.904 HKT master-replica-0 | GPU Name Persistence-M| Bus-Id Disp.A | Volatile Uncorr. ECC |
2018-12-13 12:19:44.904 HKT master-replica-0 |-------------------------------+----------------------+----------------------+
```

Figure 6 : Evaluation

After running the training, we evaluate the model and the results are :

- Average Hit@1 = 0.792
- Average Precision at equal recall rate = 0.671
- Global average precision = 0.718

3.3.Comparison of results

We decided to compare our results with the benchmark[8] made on the Youtube-8M paper for Frame level and video level features models which are : The logistic model and the Mixtures of experts for video level models and the LSTM for frame level models. We noticed that the QRNN performs better than the LSTM model and the other models. The results are summed up in the table below.

Level	Model	PERR	Hit@1
Video	QRNN(our study)	0.671	0.792
Video	Logistic regression	53.0	60.5
Video	Mixtures of Experts	55.8	63.3
Frame	LSTM	57.3	64.5

We also compared QRNN performances to the results of top performers in the 2nd edition of the Youtube-8M challenge on Kaggle [9] and with the results of a similar study on QRNNs conducted by To Isaac Zachary [10].
Some teams where from individual PhDs students, other where from big companies like Samsung. The results are summed up in the table below.

Rank	Team	Models in ensemble	GAP
1	Next top GB model	15	0.88987
2	Samsung AI Center Moscou	115	0.88729
3	PhoenixLin	3	0.88722
n/a	To Isaac Zachary	1	0.808
n/a	Our study(QRNN)	1	0.718

We can see that, after few epochs and no ensembling performed, the QRNNs performed relatively well in the challenge with a performance above 50% of GAP and not so far from top performers of the competition.
The difference can be explained by the fact that we didn't perform ensemble like most of the competitors and we also stopped the training after 3 hours due to the VM cloud utilization fees. Another reason could also be the fact that the QRNN version we used was not an official version provided by the authors of the model. Indeed we found a Keras implementation of the QRNN model that was used to compare the QRNN model training speed to the LSTM model for language understanding tasks, but this Keras version didn't fit the starter code provided by Google for the competition which is design for working with TensorFlow.
So, we decided to implement a TensorFlow version of the QRNN in order to use the starter code. We are persuaded that this TensorFlow implementation can be optimized again.

4. Further exploration

In the future, with more computation resources available, I would like to stack multiple QRNN network together, ensemble multiple models as the competitors did, train the model for a long time and make a comparison.

References

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

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

[3] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, Kevin Wilson. CNN architecture for Large-scale audio classification.

[4] Karpathy, A. (n.d.). Convolutional Neural Net- works (CNNs / ConvNets). Retrieved from http://cs231n.github.io/convolutional-networks/conv

[5] Andrej Karpathy, Justin Johnson, and Fei-Fei Li. Visualizing and understanding recurrent networks. CoRR, abs/1506.02078, 2015.

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

[7] Sepp Hochreiter and Jürgen Schmidhuber.Long short- term memory. Neural Comput., 9(8):1735–1780, November 1997.

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

[9] J. Lee, A. Natsev, W. Reade, R. Sukthankar, G. Toderici. The 2nd Youtube-8M video understanding challenge.

[10] To Isaac Zachary, HKUST student, Large-scale multi-label video classification using QRNNs, summer 2018