Master of Science in Data Science

YouTube-8M Video Understanding

Video Multi-Label Classification using Deep Learning and Decision Trees Methods

Master's Project Presentation

Prepared by:

Ehab Mohamed

Agenda

- Overview
- Dataset
- Objective
- Methodology
- Experiment
- Results
- Conclusion

Overview

Overview

- YouTube videos captures a cross-section of the society by touching all aspects of life.
- The potential of analysing and understanding large-scale videos raises the need for the project.
- With the presence of a large-scale labelled dataset and cloud computing advances, implementation of the project becomes feasible.

Overview (Cont.)

- Such research can aid in understanding various activities which are presented in videos.
- Varity of applications may get use of the research like:
  - Park Activity Analysis (i.e., analysing activities performed in park to improve area utilization and facilities),
  - Violence Activity Monitoring (i.e., monitoring violence activities performed in an area),
  - and Robot Activity Recognition (i.e., recognizing various activities by robot for better interaction).

Dataset

Youtube-8M

- The Youtube-8M dataset is introduced by Google in 2016.
- Youtube-8M is the largest and most diverse multi-label video classification dataset.
- It composed of almost 7+ million videos with about 500K hours of video.
- Each video is annotated with one or more labels from almost 4,716 labels (i.e., entities or classes).

Features

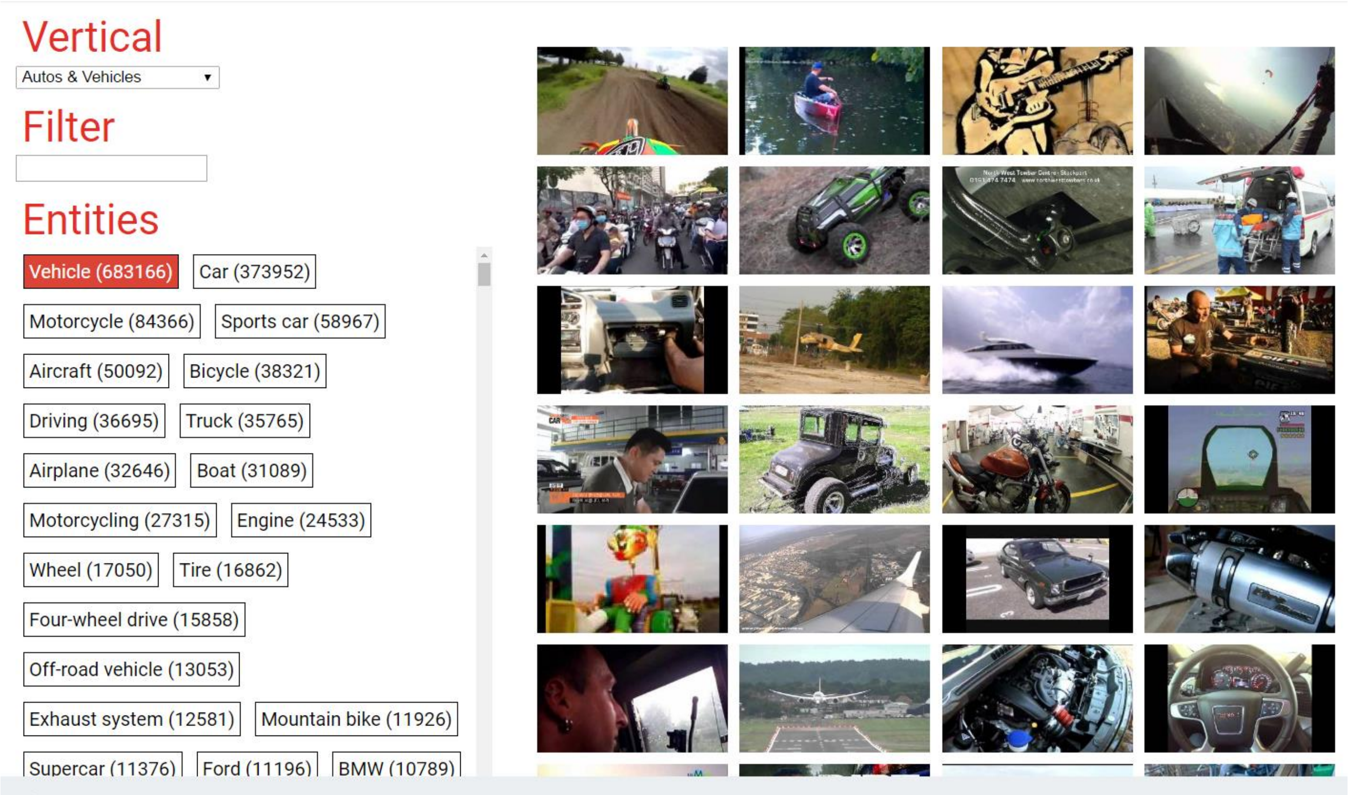
- The videos represented in the dataset were pre-processed to extract 1.6 Billion visual and 1.6 Billion audio features.

- Features are extracted at the video-level as well as frame-level granularity (at 1 frame-per-second up to the first 360 seconds).

- Visual features were extracted using Inception-V3 deep model (i.e., the publicly available Inception network trained on ImageNet).

- Audio features were extracted using a VGG-inspired acoustic model.

- Initially, each feature vector was 2048-dimensional per second of video. PCA (Principle Component Analysis) was then applied to reduce feature dimensions to 1024.



Vertical

Autos & Vehicles ▾

Filter

Entities

Vehicle (683166) Car (373952)

Motorcycle (84366) Sports car (58967)

Aircraft (50092) Bicycle (38321)

Driving (36695) Truck (35765)

Airplane (32646) Boat (31089)

Motorcycling (27315) Engine (24533)

Wheel (17050) Tire (16862)

Four-wheel drive (15858)

Off-road vehicle (13053)

Exhaust system (12581) Mountain bike (11926)

Supercar (11376) Ford (11196) BMW (10789)

Statistics

- On average, there are 3,552 training videos per entity.

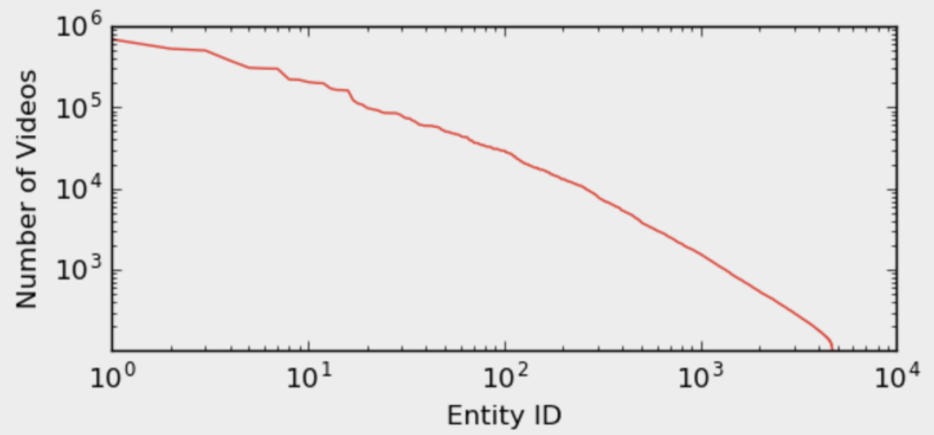
- The least number of videos per entity is 101 video examples.

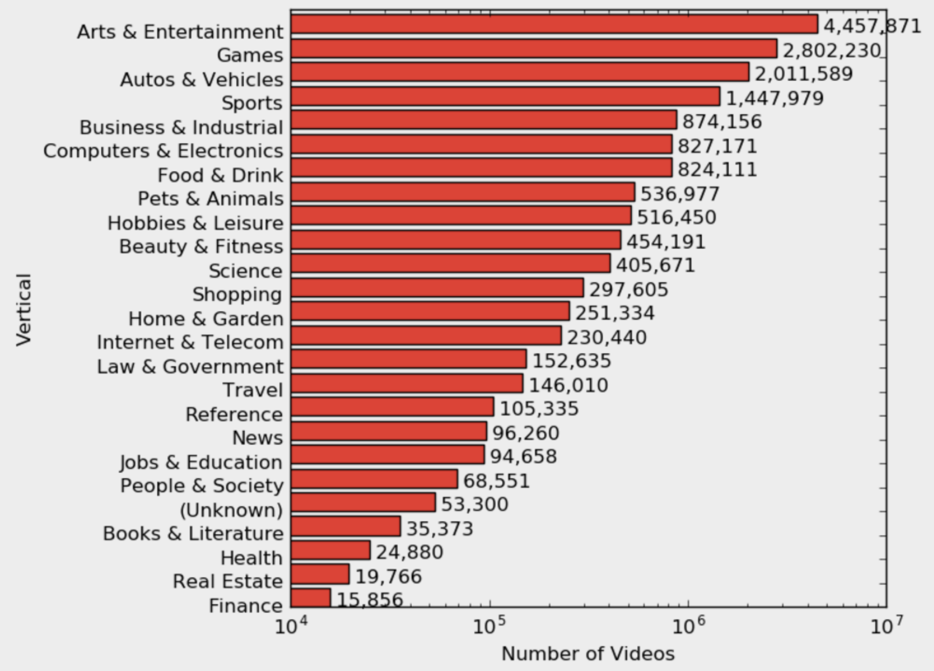
- The entity frequencies (i.e., number of videos per entity) in log-log scale, which shows a Zipf-like distribution.

- The entities are grouped into 24 high-level verticals.

- The most frequent vertical is Arts & Entertainment with more than 2.8M videos.

- The least frequent vertical is Finance with less than 14K videos.





Objective

Objective

- Examine two different types of classification models:
  - Deep Learning
    - Deep Neural Network (DNN)
    - Mixture of Experts (MoE)
  - Decision Tree
    - Extreme Gradient Boosting (XGB) with Binary Relevance problem transformation.
- DNN and MoE are examined by tuning different parameters:
  - No. of mixture of experts for MoE
  - No. of layers and neurons for DNN
  - Regularization factor
  - Learning rate
  - Dropout
  - No. of steps

Objective (Cont.)

- The XGB is implemented after performing Binary Relevance (BR) problem transformation which aims to generate a specific binary classification model for each label.
- The project focuses only on the video-level aggregated features due to the size factor and to perform quick iterations and producing results within the project's planned time frame.
- Cloud computing technology is utilized to gain high processing power in order to train and test the implemented models efficiently.

Methodology

Deep Neural Network (DNN)

- Uses multilayer perceptron with multiple hidden layers.
- Can learn more complicated functions of the data.
- Each hidden layer in the DNN combines the values in its preceding layer and learns more complicated functions of the input.
- Consists of input layer, multi hidden layers, and an output layer.
- The output layer is a logistic regression classifier with sigmoid activation function.
- The hidden units utilize the Rectified Linear Unit (RELU) which achieves sparsity and reduces the likelihood of vanishing gradient.

Hidden unit's weight:

$$z_h = RELU\left(w_h^T x\right) = \begin{cases} 0 \; if \; \left(w_h^T x\right) < 0 \\ \left(w_h^T x\right) \; if \; \left(w_h^T x\right) > 0 \end{cases}$$

Outputs:

$$y_i = sigmoid\left(v_i^T z\right) = \frac{1}{1 + exp\left[-\left(\sum_{h=1}^{H} v_{ih} z_h + v_{i0}\right)\right]}, i$$
$$= 1, \dots, K$$

The weights are obtained by minimizing the total cross-entropy error (i.e., log-loss):

$$E = \sum_{i=1}^{K} \mathcal{L}(r_i, y_i) + \lambda \|v_i\|_2^2$$

where:

$$\mathcal{L}(r, y) = -r \, \log y - (1 - r) \log(1 - y)$$

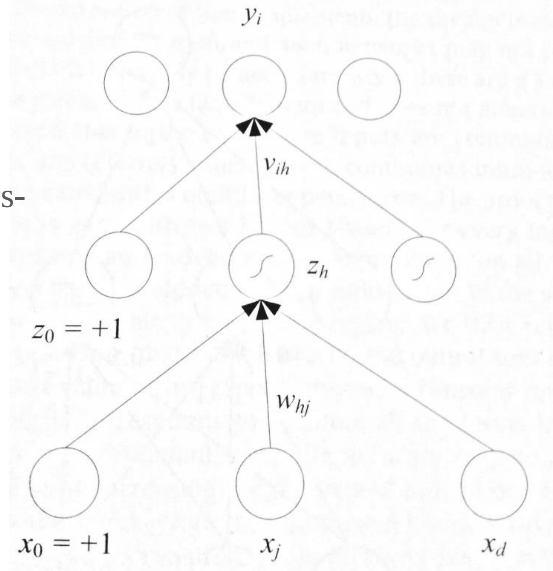
where:

$$\lambda \geq 0, \|v_i\|_2^2 = \sum_{h=1}^{H} v_{ih}^2$$

Gradient of the first-layer weights, w_{hj}, is calculated using the chain rule:

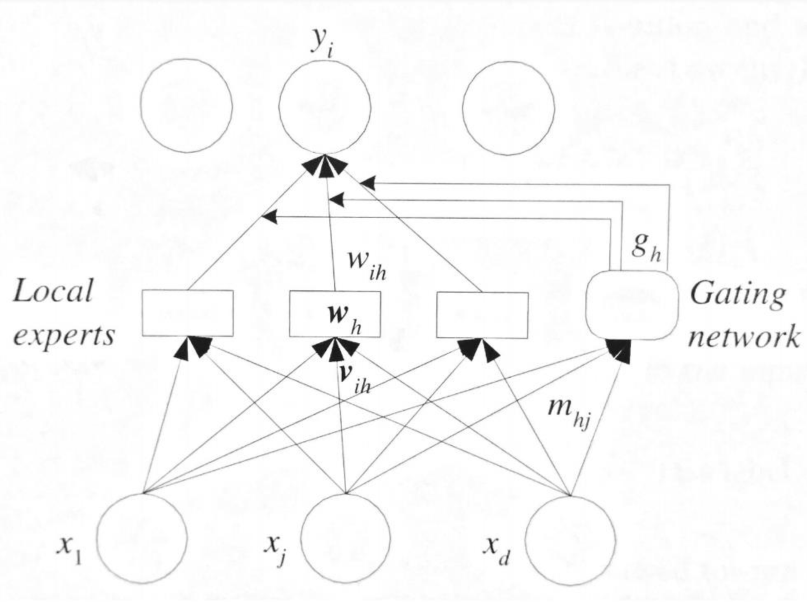
$$\frac{\partial E}{\partial w_{hj}} = \frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial z_h} \frac{\partial z_h}{\partial w_{hj}}$$

Thus, the error propagates from the output y back to the inputs (i.e., backpropagation).



Mixture of Experts (MoE)

- Can be seen as a model for combining multiple models.
- The gating network is another model which aims to determine the weight of each expert model.
- The gating network is considered as a classifier where its outputs summing to 1.



The output of the final layer is calculated as:

$$y_i^t = \sum_{h=1}^{H} w_{ih} g_h^t$$

The contribution of patch h to output i is a linear function of the input:

$$w_{ih}^t = v_{ih}^T x^t$$

The output of the gating network is a softmax function calculated as below:

$$g_h^t = \frac{exp\left(m_h^T x^t\right)}{\sum_{h'=1}^{H} exp\left(m_{h'}^T x^t\right)}$$

Extreme Gradient Boosting (XGB)

- XGB is a tree ensemble model which consists of a set of classification and regression trees (CART).

The tree ensemble model sums the prediction of multiple trees together:

$$y_i = \sum_{k=1}^{K} f_k(x_i), f_k \in F$$

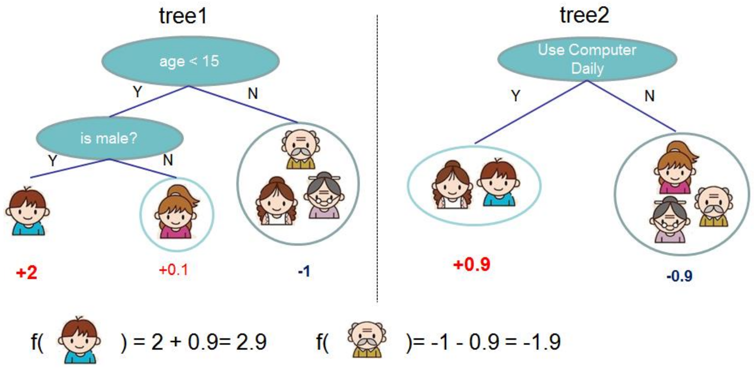
where K is the number of trees, f is a function in the functional space F, and F is the set of all possible classifiers.

Objective function :

$$E = \sum_{i=1}^{n} \mathcal{L}(r_i, y_i) + \sum_{k=1}^{K} \Omega(f_k)$$

To calculate the predictive value at step t, XGB uses an additive strategy:

$$y_i^{(0)} = 0$$
$$y_i^{(1)} = f_1(x_i) = y_i^{(0)} + f_1(x_i)$$
$$y_i^{(2)} = f_1(x_i) + f_2(x_i) = y_i^{(1)} + f_2(x_i)$$
$$\vdots$$
$$y_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = y_i^{(t-1)} + f_t(x_i)$$



In order to identify the best tree at each step, the objective function is optimized:

$$E^t = \sum_{i=1}^{n} \mathcal{L}\left(r_i, y_i^{(t)}\right) + \sum_{i=1}^{t} \Omega(f_i)$$
$$= \sum_{i=1}^{n} \mathcal{L}\left(r_i, y_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t)$$
$$+ \, constant$$

If MSE is considered to be used as the loss function, the objective function becomes in the following form:

$$E^t = \sum_{i=1}^{n} \left(r_i - \left(y_i^{(t-1)} + f_t(x_i)\right)\right)^2 + \sum_{i=1}^{t} \Omega(f_i)$$
$$E^t$$
$$= \sum_{i=1}^{n} \left[2\left(y_i^{(t-1)} - r_i\right) f_t(x_i) + f_t(x_i)^2\right] + \Omega(f_t)$$
$$+ \, constant$$

Extreme Gradient Boosting (XGB) (Cont.)

For generalization, the Taylor expansion of the loss function is taken up to the second order:

$$E^t = \sum_{i=1}^{n}\left[\mathcal{L}\left(r_i, y_i^{(t-1)}\right) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^{\,2}(x_i)\right] + \Omega(f_t) + constant$$

where g_i and h_i are defined as:

$$g_i = \partial_{y_i^{(t-1)}} \mathcal{L}\left(r_i, y_i^{(t-1)}\right), h_i = \partial^2_{y_i^{(t-1)}} \mathcal{L}\left(r_i, y_i^{(t-1)}\right)$$

After removing all the constants, the specific objective at step t becomes:

$$E^t = \sum_{i=1}^{n}\left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^{\,2}(x_i)\right] + \Omega(f_t)$$

In order to identify the complexity of the tree $\Omega(f)$, the definition of the tree $f(x)$ is refined as:

$$f_t(x) = w_{q(x)}, w \in R^T, q{:}\, R^d \to \{1,2,\dots,T\}$$

where w is the vector of scores on leaves, q is a function assigning each data point to the corresponding leaf, and T is the number of leaves. Then, the complexity is defined as:

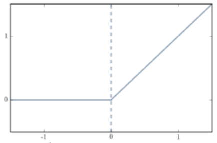
$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

Optimization Techniques

Some insights about few optimization techniques are mentioned below:

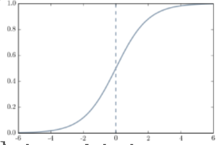
- **Backpropagation:** It is the case when the error propagates from the output y back to the inputs.
- **Rectified Activation Function (RELU):** The RELU activation function is:

$$\phi(Z) = \begin{cases} 0 \; if \; Z < 0 \\ Z \; if \; Z > 0 \end{cases}$$



- **Sigmoid Activation Function (Sigmoid):** The Sigmoid activation function is:

$$\phi(Z) = \frac{1}{1 + exp(-Z)}$$

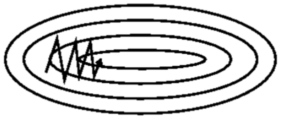


- **Regularization:** It refers to a second term that penalizes complex models with large variance as below:

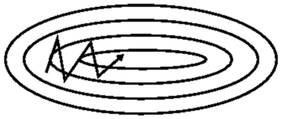
$$E' = error \; on \; data + \lambda \cdot model \; complexity$$

where λ is the weight of the penalty.

- **Adaptive Moment Optimizer (AdamOptimizer):** Adam is a computationally efficient method with little memory requirement for first-order gradient-based optimization of stochastic objective functions. Momentum is a method that helps accelerate SGD in the relevant direction and dampens oscillations. It does this by adding a fraction γ of the update vector of the past time step to the current update vector



SGD without momentum



SGD with momentum

- **Dropout:** Dropout is when the model is forced to only learn strong relationships by randomly stopping inputs to neurons. Where at random, at a percentage we set, input is set to 0.

Evaluation Function

For each video label/confidence pairs are sorted by the confidence score, then the evaluation function, which is the Global Average Precision (GAP), is applied:

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

where:

- N is the number of final predictions. If there are 20 predictions for each video, then N = 20 * [No. of Videos].
- $p(i)$ is the precision.
- $r(i)$ is the recall.

Experiments

DNN Implementation

- Utilized up to 7,000 perceptron and 5 layers including input, hidden and output layers.

- The RELU activation function was used by the input and the three hidden layers

- The Sigmoid activation function was used by the output layer.

- The model input is a matrix of the number of features, including video and audio features, multiply by the batch size (i.e., 1,152 x 1024) where the default batch size is 1,024.

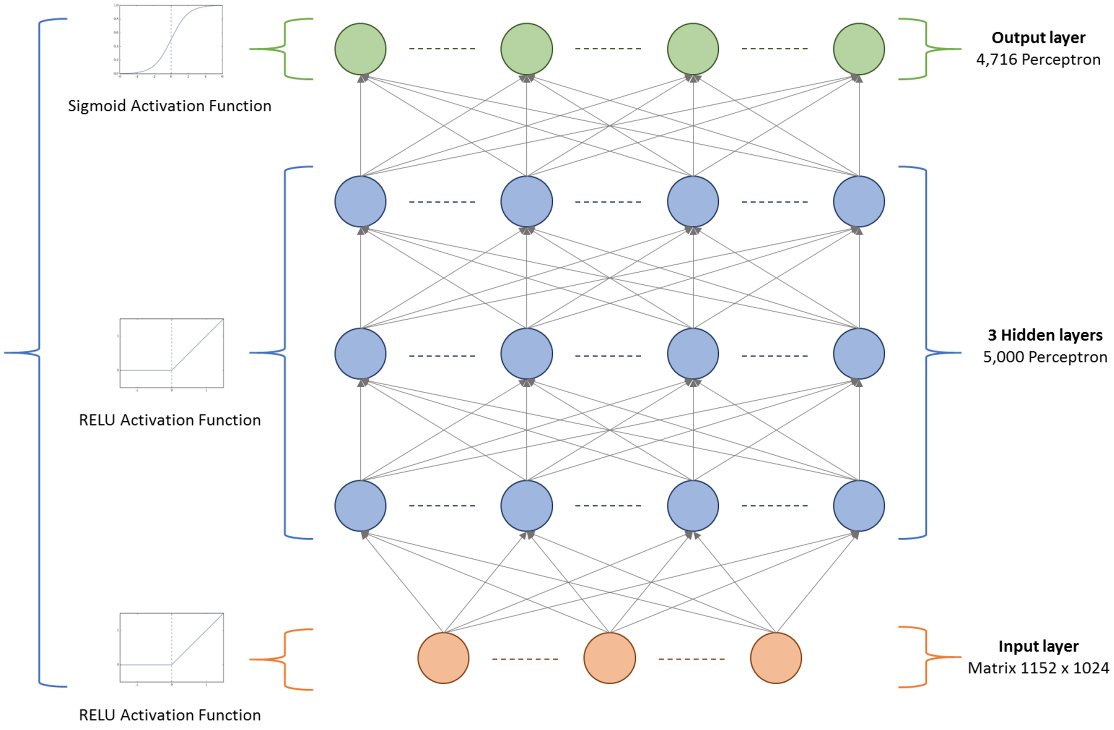
- The model number of outputs is the number of labels (4,716).

- L2 was used for all weights regularization with penalty of 1e^(-8).

- The loss function is CrossEntropyLoss, while the used optimizer is AdamOptimizer.

Default parameters:

- Weights regularizer: L2 penalty: 1e-8
- CrossEntropyLoss
- AdamOptimizer
- Loss L2 regularization weight: 1
- Learning rate: 0.01
- Learning rate decay factor: 0.95
- Learning rate decay examples: 4M



MoE Implementation

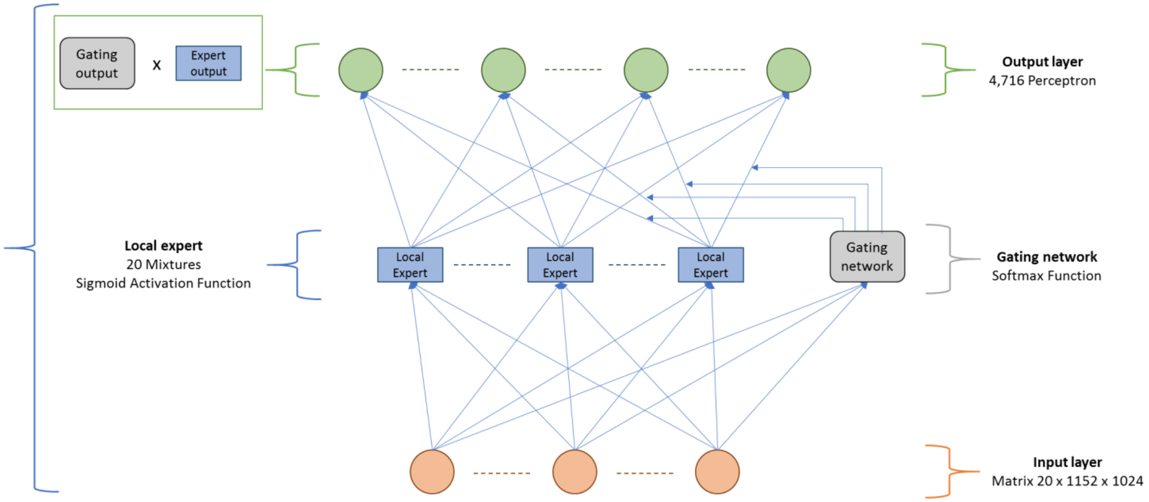
- Consists of input layer, mixture of local experts, gating network, and output layer.

- Contains up to 20 mixtures of local experts.

- The model input is a matrix of the number of features x the batch size x the number of experts.

Default parameters:

- Weights regularizer: L2 penalty: 1e-8
- CrossEntropyLoss
- AdamOptimizer
- Loss L2 regularization weight: 1
- Learning rate: 0.01
- Learning rate decay factor: 0.95
- Learning rate decay examples: 4M



Local expert
20 Mixtures
Sigmoid Activation Function

Output layer
4,716 Perceptron

Gating network
Softmax Function

Input layer
Matrix 20 x 1152 x 1024

XGB Implementation

- Implementation has taken the approach of binary relevance (BR) transformation

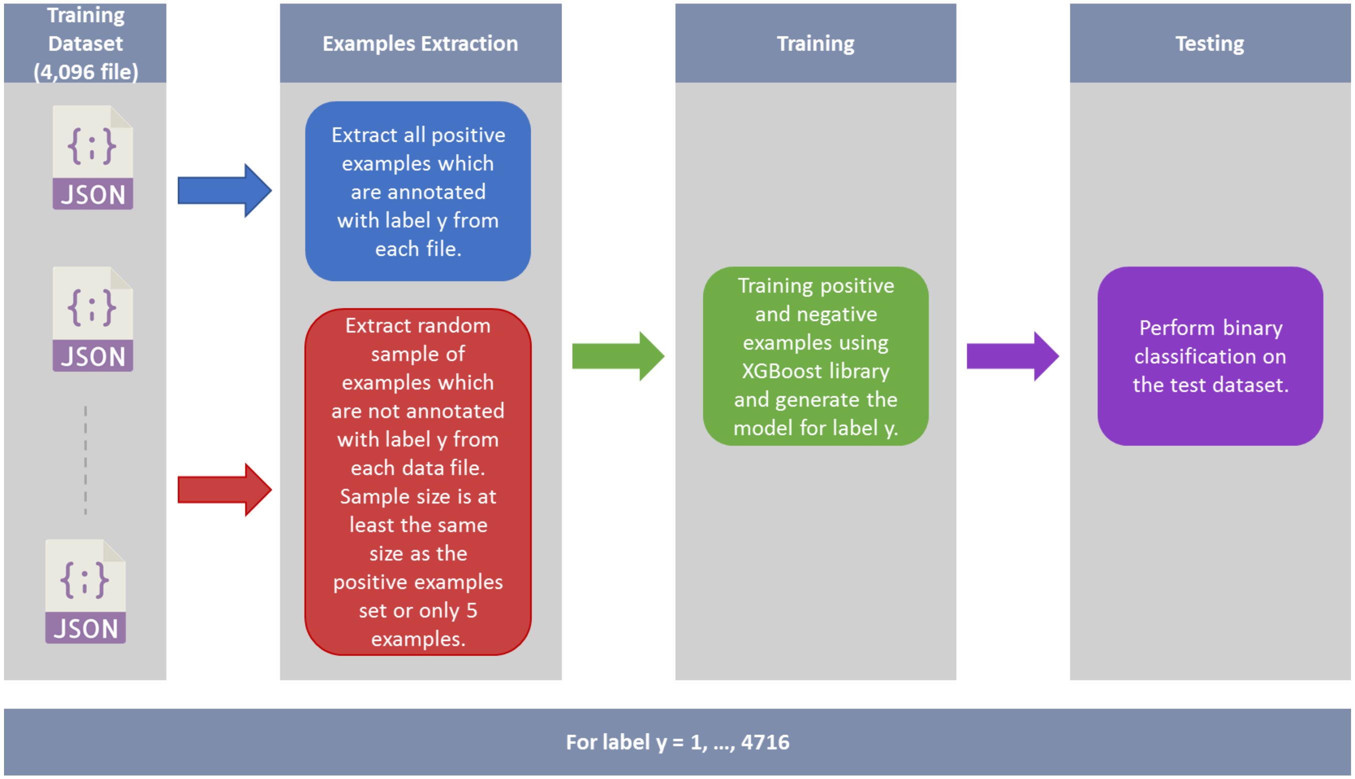
- The multi-label classification problem is split into multiple binary classification problem.

- A binary XGB model is trained to classify each label separately to form at the end an ensemble consists of 4,716 (i.e., total number of labels) relevance classifier.

- All trained classifiers form together the anticipated multi-label classification model.

- Each XGB model is trained through 30 iterations using a 'binary:logistic' classifier and CrossEntropy loss function.

- One challenge is that the dataset is unbalanced where labels do not have the same distribution.



Training Dataset (4,096 file)

Examples Extraction

Extract all positive examples which are annotated with label y from each file.

Extract random sample of examples which are not annotated with label y from each data file. Sample size is at least the same size as the positive examples set or only 5 examples.

Training

Training positive and negative examples using XGBoost library and generate the model for label y.

Testing

Perform binary classification on the test dataset.

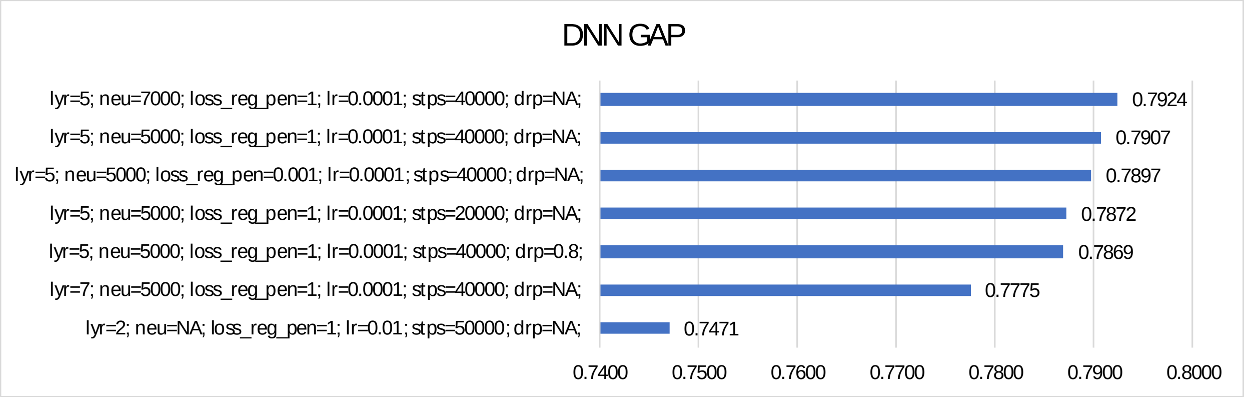
For label y = 1, ..., 4716

Results

DNN Results

- There are main six parameters involved in the training process which are No. of Layers, No. of Neurons, Loss Regularization Penalty, Learning Rate, Max steps, and the Dropout Keeping Rate.
- The model trained with 5 layers, 7K neurons (i.e., perceptrons), loss regularization penalty of 1, learning rate of 0.0001, 40K steps, and without involving the dropout is the best performer model with a GAP of 0.79238.

No. of Layers	No. of Neurons	Loss Regularization Penalty	Learning Rate	No. of Steps	Dropout	GAP
5	7000	1	0.0001	40000	NA	0.79238
5	5000	1	0.0001	40000	NA	0.79071
5	5000	0.001	0.0001	40000	NA	0.78973
5	5000	1	0.0001	20000	NA	0.78723
5	5000	1	0.0001	40000	0.8	0.78694
7	5000	1	0.0001	40000	NA	0.77750
2	NA	1	0.01	50000	NA	0.74708


DNN GAP

DNN Results (Cont.)

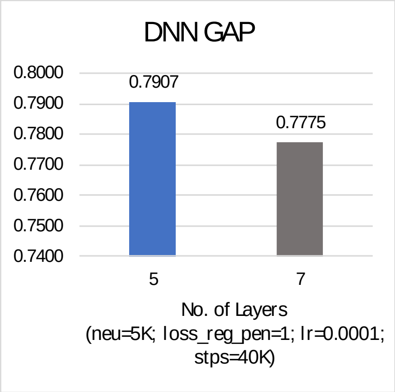
- When the no. of layers increased from 5 to 7 the GAP of the DNN model decreased.

- Decreasing the loss regularization penalty had a negative impact on the GAP score.

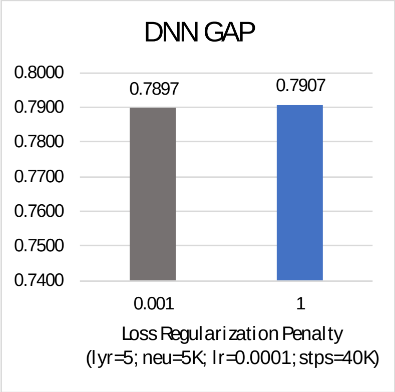
- Utilizing the dropout technique, with 0.8 keeping ratio, didn't show any improvement in the GAP score.

- Training the model more steps showed slight improvement in the GAP score.

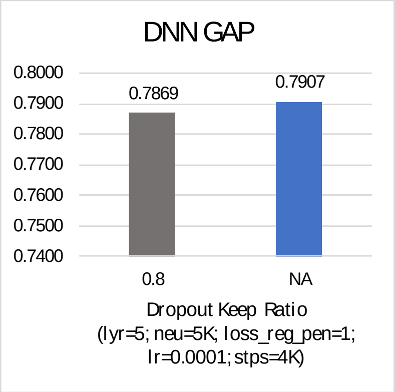
- Finally, increasing no. of neurons in the hidden layer from 5K to 7K shows a slight improvement in the performance.



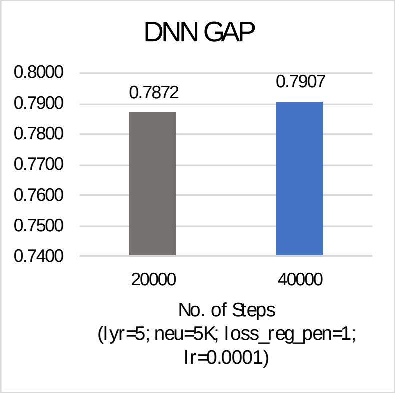
a. DNN GAP by no. of layers



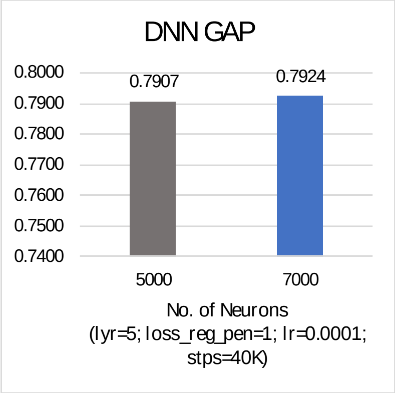
b. DNN GAP by loss regularization penalty



c. DNN GAP by the dropout keep ratio



d. DNN GAP by no. of steps

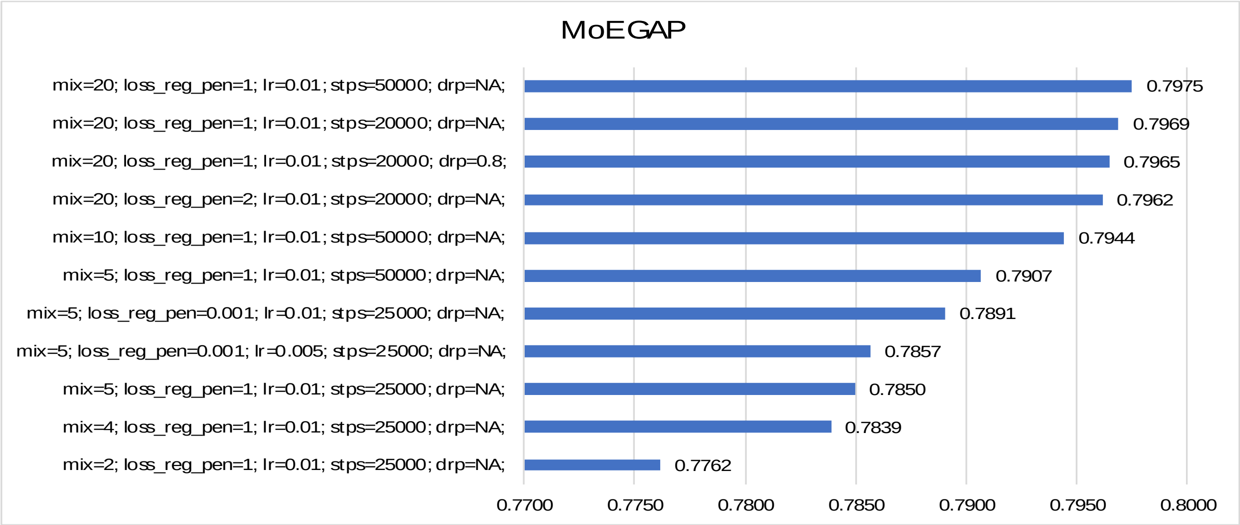


e. DNN GAP by no. of neurons

MoE Results

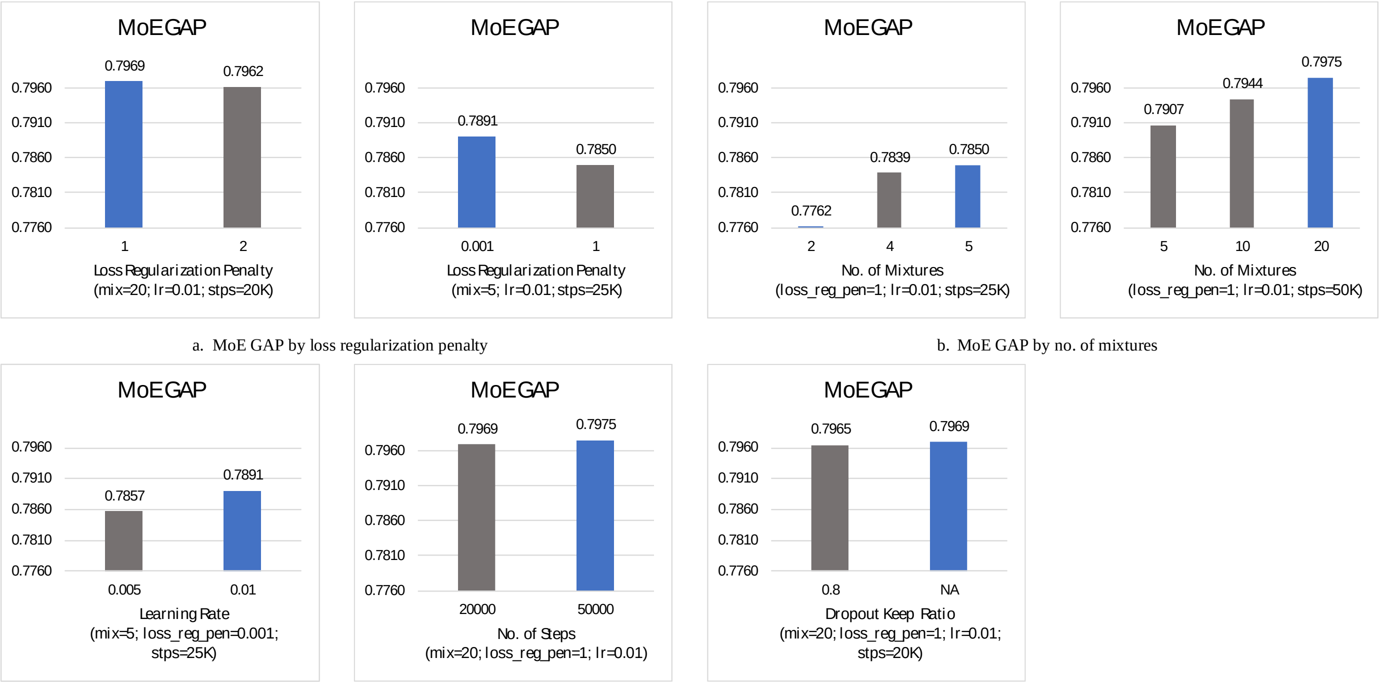
- The parameters are No. of Mixtures, Loss Regularization Penalty, Learning Rate, No. of Steps, and the Dropout Keep Ratio.
- The best performer model is the one with 20 mixtures, loss regularization penalty of 1, learning rate of 0.01, 50K training steps, and without utilizing the dropout technique.

No. of Mixtures	Loss Regularization Penalty	Learning Rate	No. of Steps	Dropout	GAP
20	1	0.01	50000	NA	0.79751
20	1	0.01	20000	NA	0.7969
20	1	0.01	20000	0.8	0.79647
20	2	0.01	20000	NA	0.79616
10	1	0.01	50000	NA	0.79443
5	1	0.01	50000	NA	0.79069
5	0.001	0.01	25000	NA	0.78905
5	0.001	0.005	25000	NA	0.78567
5	1	0.01	25000	NA	0.78497
4	1	0.01	25000	NA	0.78388
2	1	0.01	25000	NA	0.77618



MoE Results (Cont.)

- Increasing the loss regularization penalty from 1 to 2 reduced the GAP score, whereas reducing it from 1 to 0.001 shows slight improvement in the score.
- Increasing the no. of mixtures from 2 to 20 shows significant improvement in the GAP score.
- Increasing the learning rate from 0.005 to 0.01 shows small improvement in the score.
- Training the model more steps has a slight positive impact on the score.
- Using the dropout technique didn't show any improvement in the GAP score.



a. MoE GAP by loss regularization penalty

b. MoE GAP by no. of mixtures

c. MoE GAP by learning rate

d. MoE GAP by no. of steps

e. MoE GAP by dropout keep ratio

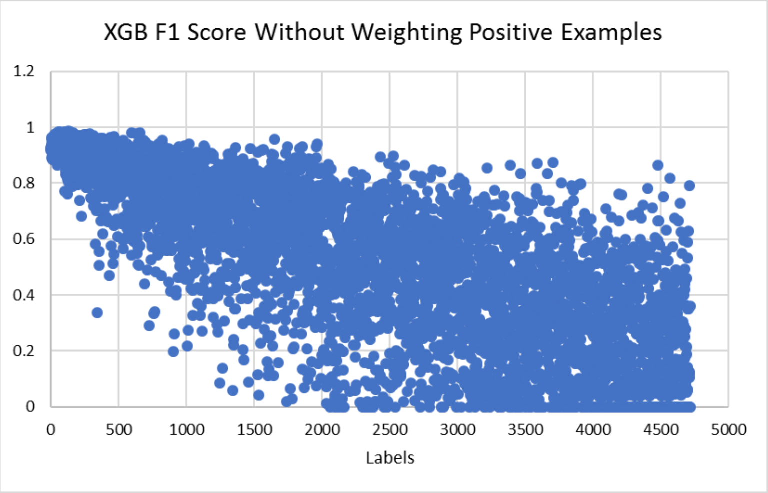
DNN and MoE Averaging

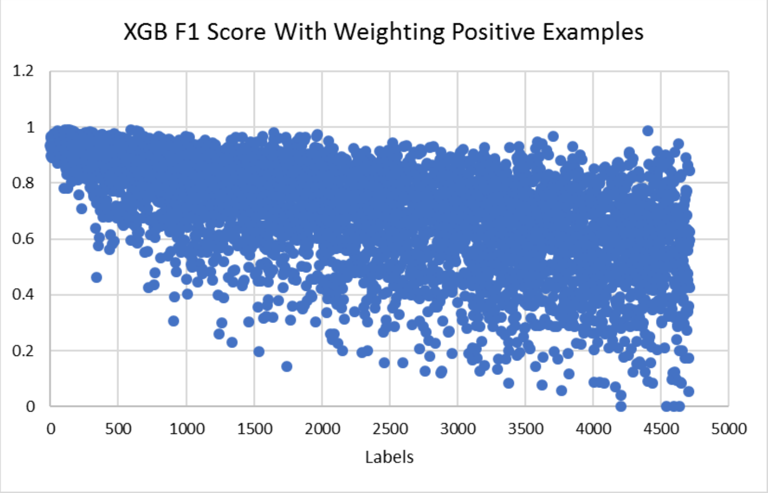
By performing weighted averaging on the best models of DNN and MoE, a new GAP score of 0.80961 was achieved.

The predictions of the best model from DNN was multiplied by 40%, whereas the predictions of the best model from MoE was multiplied by 60%, then averaging is taken to form the highest score in the experiments.

XGB Results

- Due to the complexity of forming and training the ensemble of binary classifiers (i.e., a classifier for each label) within the project time frame, the ensemble was trained twice only.
- In the first training experiment (i.e., first ensemble), no weights were assigned to the positive extracted examples to mitigate the impact of the unbalanced data, whereas, in the second training experiment (i.e., second ensemble) weights were assigned.
- After training both ensembles, they have been examined against the test dataset.
- The results show poor performance of the two trained models, however, the second ensemble (i.e., with weights assigned) is slightly better.
- The first trained ensemble scored 0.41993 GAP, while the second trained ensemble scored 0.42374 GAP.





Results Summary

- The best score is formed by averaging both the DNN and MoE best models.
- The XGB did not perform well and the implementation of the binary relevance may be revised to ensure its feasibility.

Model	Parameters/Description	GAP
Weighted Average DNN & MoE	DNN x 40% + MoE x 60%	0.80961
MoE	mix=20; loss_reg_pen=1; lr=0.01; stps=50000; drp=NA;	0.79751
DNN	lyr=5; neu=7000; loss_reg_pen=1; lr=0.0001; stps=40000; drp=NA;	0.79238
XGB	With weighting positive examples	0.42374

Conclusion

Conclusion

- Having a large-scale dataset like YouTube-8M provided a good chance to implement and examine two deep learning models (i.e., DNN and MoE) and one decision tree model (i.e., XGB).
- No. of expert mixtures in the MoE as well as no. of neurons in the DNN played an important role in improving the performance of the models.
- Loss regularization penalty was very crucial in adjusting the models' generalization.
- Although, dropout didn't provide any improvement, its implementation and ratio might be revisited for adjustment.

Conclusion (Cont.)

- Increasing no. of steps slightly improved the performance of the models, however, stopping models training in the suitable step may mitigate overfitting and improve performance.
- The XGB BR implementation with the current limited resources and within the project timeframe was very basic.
- Due to such unbalanced large-scale dataset, the XGB BR success possibility is arguable.
- However, training a single XGB model on the Youtube-8M dataset may also be investigated in the future.

Thank you.