Tutorial 3: Project 2 / MTL

2020.11.13

Qizhi Li

Outline

- Multi-label Classification on YouTube-8M
- Introduction of multi-task learning

Multi-label Classification on YouTube-8M

- Task
- Dataset
- MAP@K
- Submission Requirements
- Demo

Task

Learn

Input (128+1024)

Labels

American football

Festival

Gym

Weight training

Topic

Sports

Arts & Entertainment

Beauty & Fitness

The category of the label is vocabulary.csv

Predict

Gym

American football

Video game

YouTube 8M

Dataset Explore Download Workshop

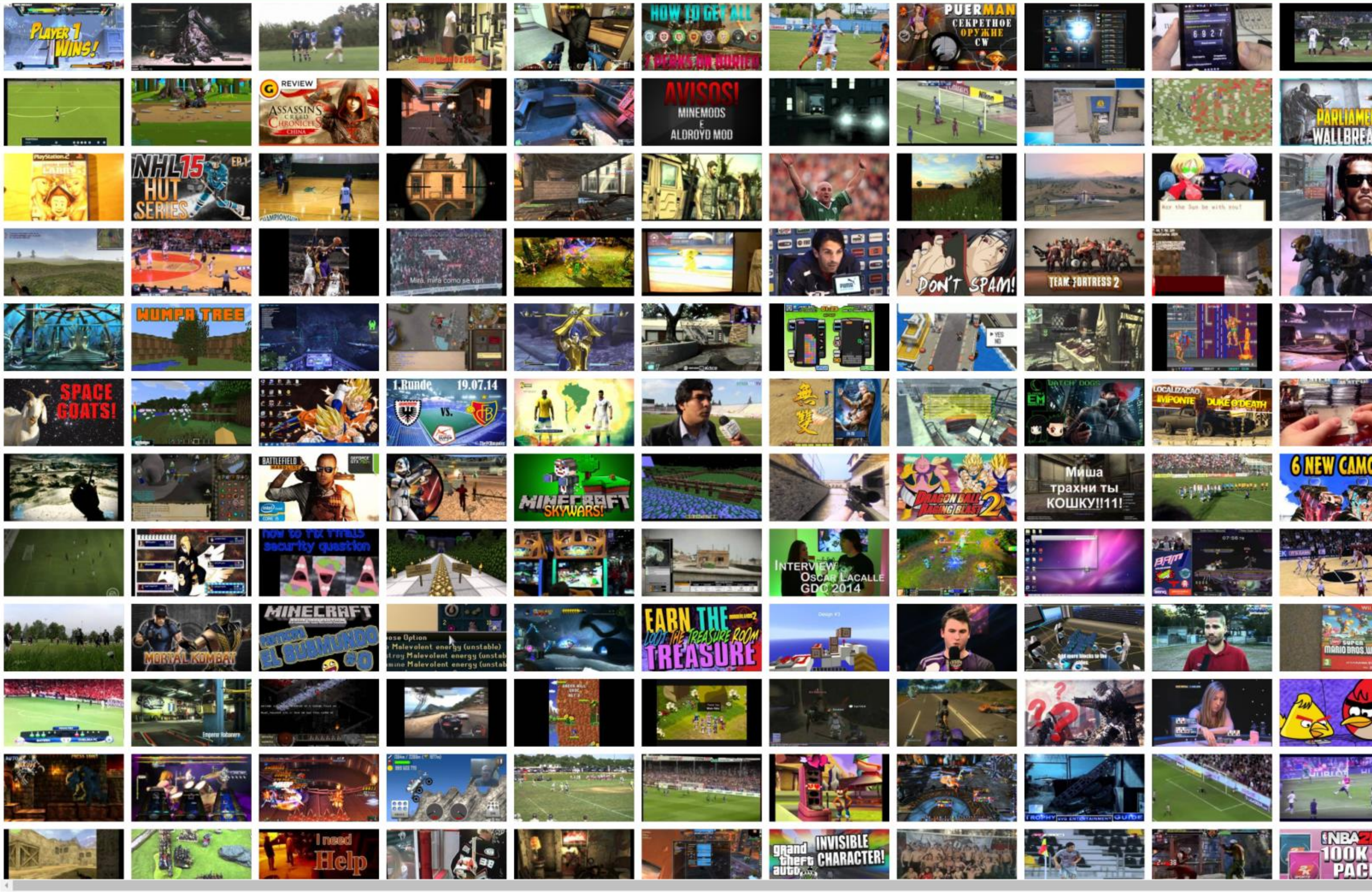
Vertical

All

Filter

Entities

Games (788288) Video game (539945)
Vehicle (415890) Concert (378135)
Musician (286532) Cartoon (236948)
Performance art (203343) Car (200813)
Dance (181579) Guitar (156226)
String instrument (144667) Food (135357)
Football (130835) Musical ensemble (125668)
Music video (116098) Animal (107788)
Animation (98140) Motorsport (93443)
Pet (90779) Racing (84258) Recipe (75819)
Mobile phone (72911) Cooking (71218)
Smartphone (64884) Gadget (64452)
Trailer (59695) Toy (58720)
Minecraft (57801) Drums (55597)
Cuisine (55411) Piano (55201)
Motorcycle (54950) Dish (54730)
Drums (54195) Acoustic guitar (51712)
Action-adventure game (51597)
Call of Duty (49465) Electric guitar (47982)
Drummer (45885) Cosmetics (43039)
Keyboard (41496) Choir (38652)
Strategy video game (36451) Fishing (36051)
Aircraft (35779) Train (35331)
Airplane (35170) Pianist (34650)



TFrecord

1. Many tf.Example is store in one Tfrecord file.
2. TF Example:
    a. As {"string": tf.train.Feature}
    b. Type of tf.train.Feature
        - tf.train.BytesList
        - tf.train.FloatList
        - tf.train.Int64List
3. Way to load
    a. Tf tutorial
    b. Following the demo

id
(bytes)

labels
(List of int)

mean_rgb
(1024)

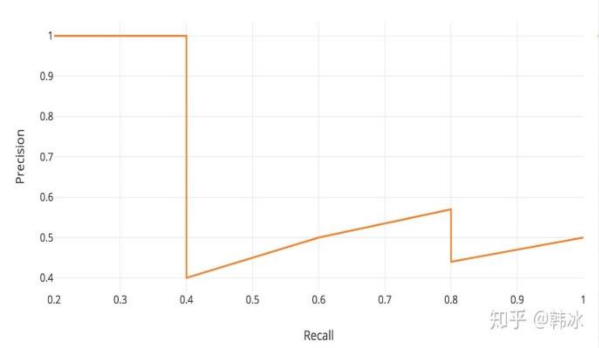
mean_audio
(128)

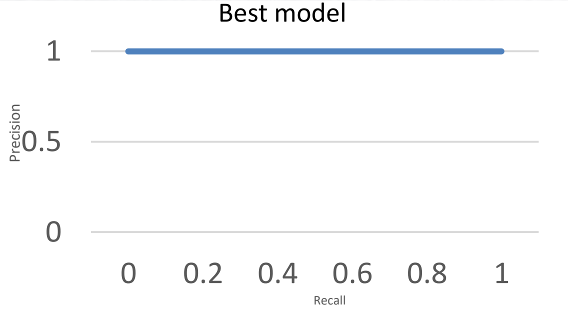
Tf.Example

```
features: {
  feature: {
    key  : "id"
    value: {
      bytes_list: {
        value: (Video id)
      }
    }
  }
  feature: {
    key  : "labels"
    value: {
      int64_list: {
        value: [1, 522, 11, 172]  # label list
      }
    }
  }
  feature: {
    # Average of all 'rgb' features for the video
    key  : "mean_rgb"
    value: {
      float_list: {
        value: [1024 float features]
      }
    }
  }
  feature: {
    # Average of all 'audio' features for the video
    key  : "mean_audio"
    value: {
      float_list: {
        value: [128 float features]
      }
    }
  }
}
```

MAP@K

1. True label: list of label {3, 4, 34, 2781, 3764}
2. Label with confidence {1:0.23, 2:0.43, 3:0.87, 4:0.86, 5:0.30, …}
3. Prediction label {4, 3, 139, 188, 215, 34, 2781, 2208, 40, 3764} with confidence decreasing.
4. Intuition:
    a. Better model will predict the true labels at higher rank.
    b. In one prediction, the recall is bigger, the precision is always smaller.





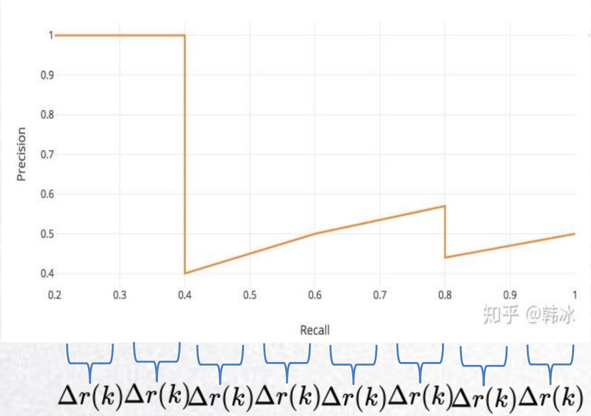
{3, 34, 2781, 4, 3764, 215, 873, 2208, 40, 3764, …}

MAP@K

1. Definition

$$\text{AveP} = \sum_{k=1}^{K} P(k)\Delta r(k)$$

$$\text{MAP} = \frac{\sum_{q=1}^{Q} \text{AveP(q)}}{Q}$$



Where K is the number of labels you output (recommend)
P(k) and r(k) is the precision and recall calculated by the subset **from rank 1 through k**;
q is one query, here it is the list containing true labels of video；
Q is the number of queries, here it is number of videos.

@ K: Evaluate on only the first K output (recommend).

AveP@K

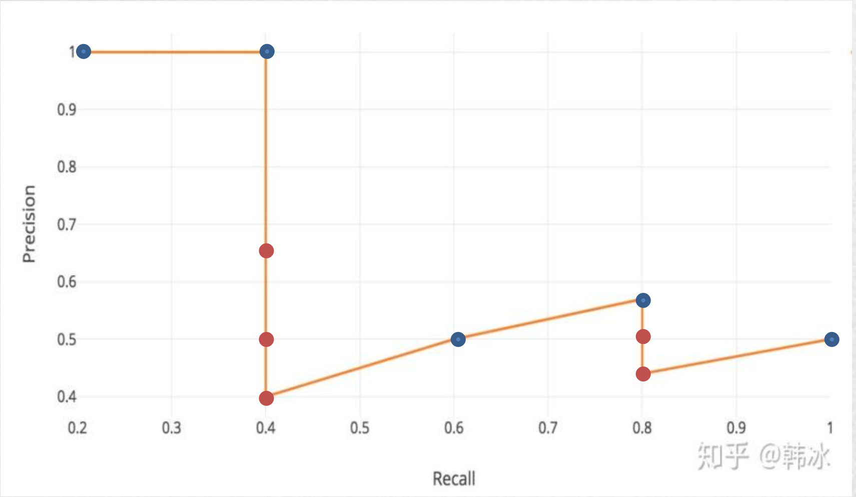
Actual label {3, 4, 34, 2781, 3764}

Prediction {4, 3, 139, 188, 215, 34, 2781, 2208, 40, 3764}

1. Calculate AveP

   a. Intuition: The area under the Precision-recall

$$AveP = \int_0^1 p(r)dr$$



Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0	0.4
3	False	0.67	0.4
4	False	0.5	0.4
5	False	0.4	0.4
6	True	0.5	0.6
7	True	0.57	0.8
8	False	0.5	0.8
9	False	0.44	0.8
10	True	0.5	1.0 知乎 @韩冰

{T, T, F, F, F, T, T, F, F, T}

P={1/1, 2/2, 2/3, 2/4, 2/5, 3/6, 4/7, 4/8, 4/9, 5/10}

R={1/5, 2/5, 2/5, 2/5, 2/5, 3/5, 4/5, 4/5, 4/5, 5/10}

P(k) and r(k) is the precision and recall calculated by the subset **from rank 1 through k**;

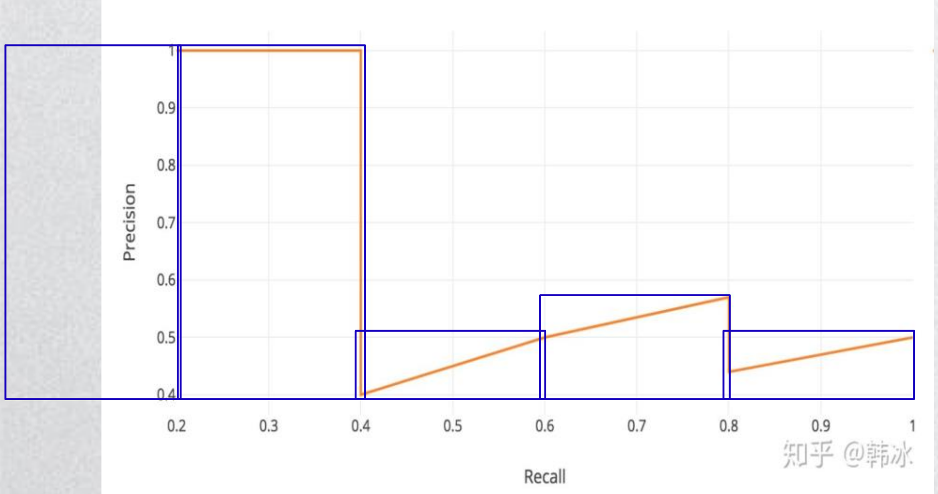
AveP@K

1. Calculate AveP
   a. Smooth: The area under the Precision-recall

Actual label {3, 4, 34, 2781, 3764}

Prediction {4, 3, 139, 188, 215, 34, 2781, 2208, 40, 3764}

$$\text{AveP} = \sum_{k=1}^{K} P(k)\Delta r(k)$$



Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0	0.4
3	False	0.67	0.4
4	False	0.5	0.4
5	False	0.4	0.4
6	True	0.5	0.6
7	True	0.57	0.8
8	False	0.5	0.8
9	False	0.44	0.8
10	True	0.5	1.0 知乎 @韩冰

{T, T, F, F, F, T, T, F, F, T}

P={1/1, 2/2, 2/3, 2/4, 2/5, 3/6, 4/7, 4/8, 4/9, 5/10}

R={1/5, 2/5, 2/5, 2/5, 2/5, 3/5, 4/5, 4/5, 4/5, 5/5}

$$\text{AveP} = \sum_{k=1}^{K} P(k)\Delta r(k) = 0.714$$

P(k) and r(k) is the precision and recall calculated by the subset **from rank 1 through k**;

Example of AP@10

1. In our question MAP@10

Actual label {3, 4, 9, 34}
Prediction label {3, 4, 139, 188, 34, 215, 2781, 2208, 40, 3764}

$$\operatorname{AveP} = \sum_{k=1}^{K} P(k)\Delta r(k)$$

Example of AP@10

$$\text{AveP} = \sum_{k=1}^{K} P(k)\Delta r(k)$$

1. In our question MAP@10

Actual label {3, 4, 9, 34}
Prediction label {3, 4, 139, 188, 34, 215, 2781, 2208, 40, 3764}

$$\begin{aligned}\text{AveP} &= \sum_{k=1}^{n} P(k)\Delta r(k) \\ &= 1/1 \times 1/4 \\ &+ 2/2 \times 1/4 \\ &+ 3/5 \times 1/4 \\ &= 0.65\end{aligned}$$

Rank	**Prediction**	**Like?**	**Precision**	**Recall**	$\Delta Recall$
1	3	Yes	1/1	1/4	1/4
2	4	Yes	2/2	2/4	1/4
3	139		2/3	2/4	0
4	188		2/4	2/4	0
5	34	Yes	3/5	3/4	1/4
6	215		3/6	3/4	0
7	2781		3/7	3/4	0
8	2208		3/8	3/4	0
9	40		3/9	3/4	0
10	3764		3/10	3/4	0

2. Kaggle Example

Assignment Requirements

1. Output: Top 10 label with highest predicted confidence (**separated by spaces**) descending order of confidence.

2. Model: Any DNN architecture. (CNN, RNN, GCN, etc)

3. You can use subset or all the following packages.

   a. Python 3.6 or Python 3.7

   b. Numpy

   c. Pandas

   d. TensorFlow-gpu 1.15.0 (or TensorFlow-cpu 1.15.0)

   e. PyTorch 1.1.0

   f. Keras 2.2.4

Dataset

1. Video-level feature dataset

   a. training data (3,844 shards, 3,888,919 videos, 17.0GB)

   b. Validation data (3,844 shards, 1,112,356 videos, 4.87GB)

   No Kaggle.
   Predict all videos in training set and validation set (21.9GB in total).
   We will randomly select data from your predictions to score.

```
# create the folder as “train” and “validation”
# in “train” folder
curl data.yt8m.org/download.py | partition=2/video/train mirror=asia python
# in “validation” folder
curl data.yt8m.org/download.py | partition=2/video/validate mirror=asia python
```

Output and Marking

1. Evaluation will belong to 3 parts:
   a. 50 shards in train0000.tfrecord~train0199.tfrecord
   b. 30 shards in validate0000.tfrecord~validate0099.tfrecord
   c. 20 shards from all remaining in the training and validation set

80% of the data in a small range guaranteed the lower bound of your score.

2. Data to be predicted
   a. put all the training and validation data in one folder named "train_validation/"
3. test.py
   a. first sort all the tfrecord files in "train_validation/" (21.9GB)
   b. must use relative path "train_validation/" in your submitted test.py file to access the data.
4. Label: 10 labels should be arranged in decreasing order (**separated by spaces**)

Submission Requirement

1. To Canvas (Less than 600MB)
   a. train.py
   b. test.py
   c. output_ student_id.txt
   d. Weight of your models (Folder)
   e. Report (in paper format, include but not limited)
      - Title
      - Introduction
      - Related Work (optional)
      - Methods
      - Experiments (with possible discussions)
      - Conclusion
      - The role and specific contributions of each group member

      There is no page limit.
   f. Presentation slides (7-minute presentation at Dec 24, Thursday)

```
├──518XXXXXXXXX.zip
│    │ —output_518XXXXXXXXX.csv
│    │ —test.py
│    │ —train.py
│    │ —Report.pdf
│    │ —slides_leader_student_id.pptx
│    │ —other files
│    └──weights (Take TensorFlow as an example)
│          │ —checkpoint
│          │ —model.data-00000-of-00001
│          │ —model.index
│          │ —model.meta
```

NeurIPS 2020 format is recommended

Demo

1. The entire folder demo: CS410_2020_fall_project_2

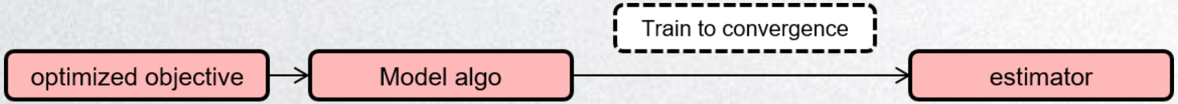
Introduction of Multi-task Learning

- Introduction/Motivation
- General Approach
- Specific Related Works

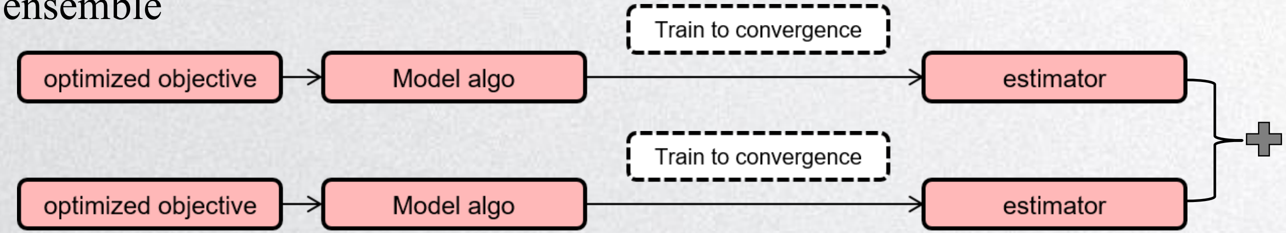
  MMoE/ SNR/ PLE
- Implementation details

Introduction

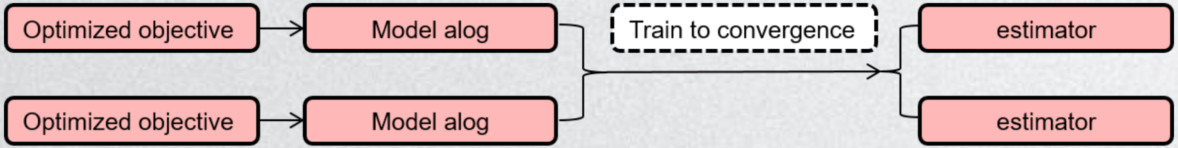
1. Single Model



2. Model ensemble



3. Multi-task Learning



Scenarios

1. Finance or economics forecasting
    - Predict the value of many possibly related indicators
2. Bioinformatics
    - predict symptoms for multiple diseases simultaneously
3. Self-driving car
    - predict different characteristics of the road as auxiliary tasks for predicting the steering direction
4. Recommendation system
    - Predictions on multi business KPIs, CVR, CTR

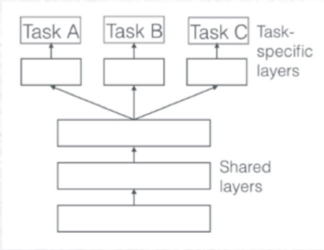
General Approach

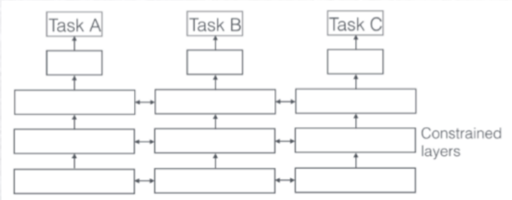
1. Deep Model
   a. Hard parameter sharing
      - Sharing the hidden layers between all tasks
      - keeping task-specific output layers

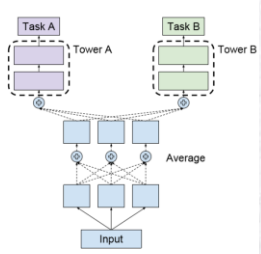
   b. Soft parameter sharing
      - Each task has its own parameters while the distance
      - between parameters is regularized to be similar

   c. Learning what to share
      - Determine sharing parts and manners between
      - tasks without prior information

2. Linear/Shallow Model (not in here)
   a. Learning task relationships
      - Tasks are in cluster/tree/graph/hierarchy structure

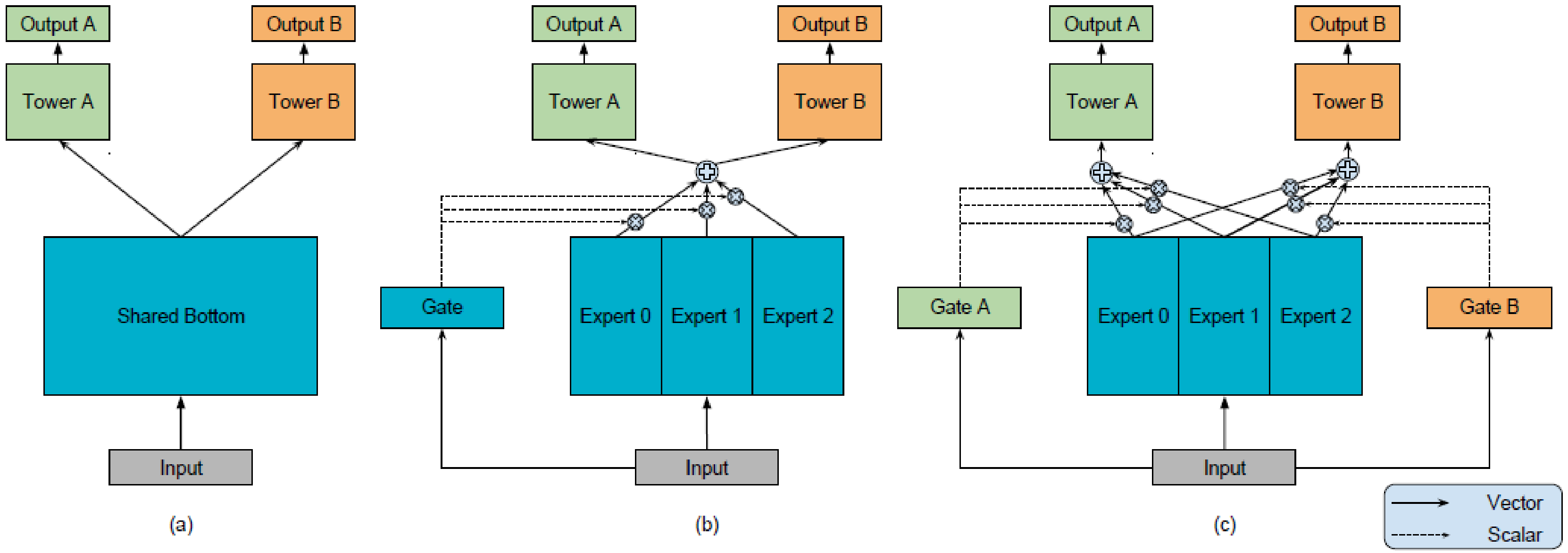






Specific Related Works

1. MMoE



Shared Bottom

OMoE

All tasks share a set of Gates

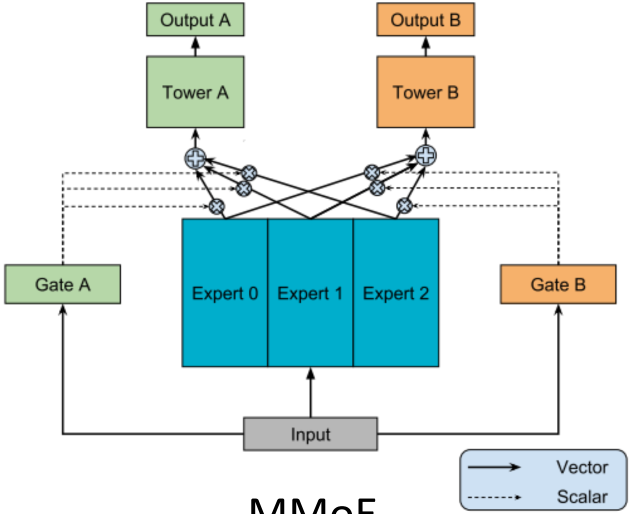
MMoE

Each task uses a set of Gates

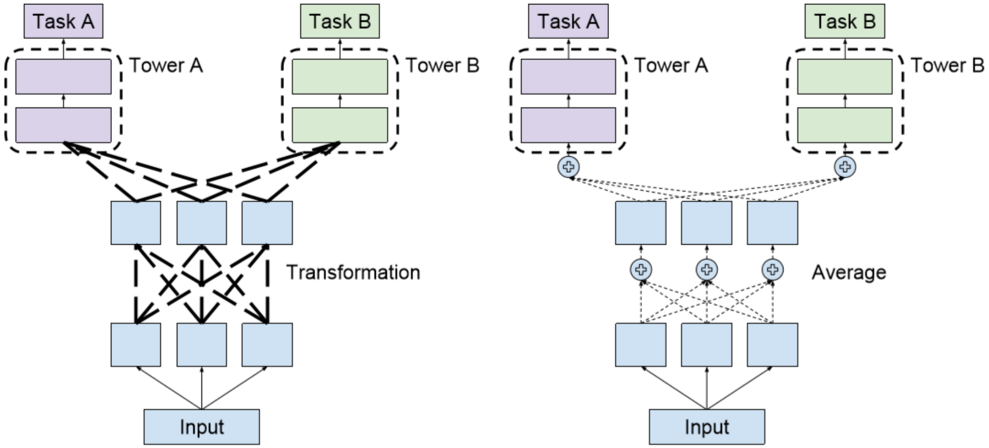
Ma, Jiaqi, et al. "Modeling task relationships in multi-task learning with multi-gate mixture-of-experts." *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2018.

Specific Related Works

2. SNR



MMoE



(b) SNR-Trans

(c) SNR-Aver

$$f^k(x) = \sum_{i=1}^{n} g_i^k(x) f_i(x)$$

Where $g_i^k(\mathrm{x}) = \mathrm{softmax}(\mathrm{W}_{gk}\mathrm{x})$

$$f^k(x) = \sum_{i=1}^{n} g_i^k(x) f_i(x)$$

Where$g_i^k(\mathrm{x}) = \mathrm{z}_{\mathrm{ik}} W_{ik}$

$$f^k(x) = \sum_{i=1}^{n} g_i^k(x) f_i(x)$$

Where$g_i^k(\mathrm{x}) = \mathrm{z}_{\mathrm{ik}}$

binary coding variables $z_i \in \{0,1\}$

Ma, Jiaqi, et al. "Snr: Sub-network routing for flexible parameter sharing in multi-task learning." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 2019.

SNR two types of connections

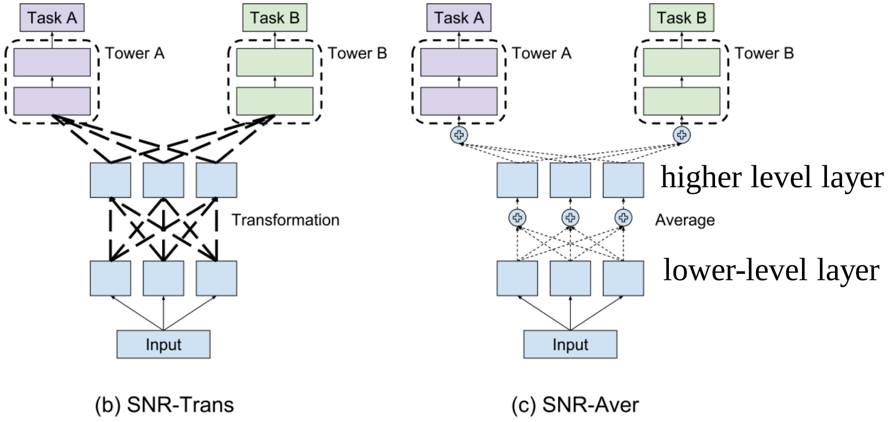
- SNR-Trans use matrix transformations

$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} z_{11}\boldsymbol{W_{11}} & z_{12}\boldsymbol{W_{12}} & z_{13}\boldsymbol{W_{13}} \\ z_{21}\boldsymbol{W_{21}} & z_{22}\boldsymbol{W_{22}} & z_{23}\boldsymbol{W_{23}} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$$

- SNR-Aver use identity matrix

$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} z_{11}\boldsymbol{I_{11}} & z_{12}\boldsymbol{I_{12}} & z_{13}\boldsymbol{I_{13}} \\ z_{21}\boldsymbol{I_{21}} & z_{22}\boldsymbol{I_{22}} & z_{23}\boldsymbol{I_{23}} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$$

binary coding variables $z_i \in \{0,1\}$



  - u_1, u_2, u_3 be the outputs of the lower-level
  - v_1, v_2 be the inputs of the higher-level
  - W_{ij}: transformation matrix from the j th lower-level network to the i th higher level network.
  - I_{ij} : identity matrix for all i, j

- SNR-Trans has more model parameters in the connection
- SNR-Aver has more budget of model parameters in the sub-networks.

Specific Related Works

3. CGC/PLE

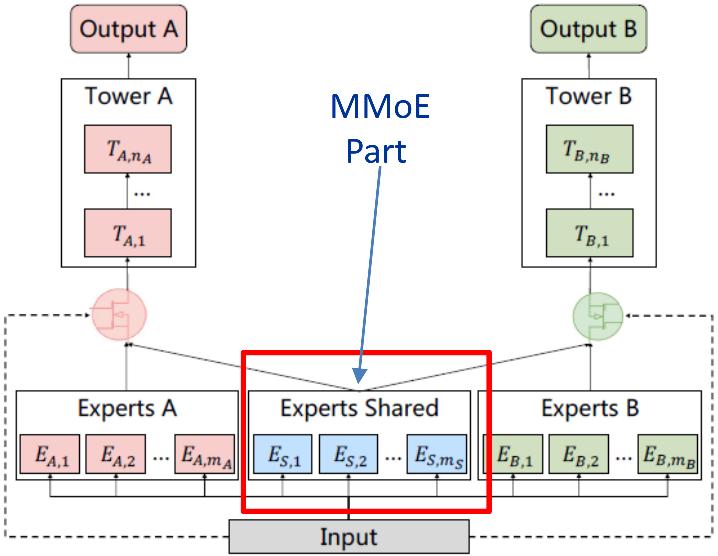
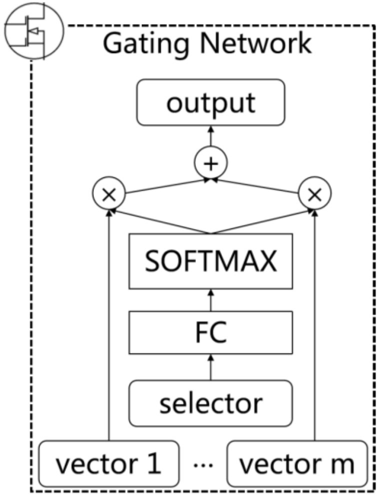


Figure 4: Customized Gate Control (CGC) Model



Gate: Only use the output of the **shared expert** and the **expert exclusive to the task.**

Tang, Hongyan, et al. "Progressive Layered Extraction (PLE): A Novel Multi-Task Learning (MTL) Model for Personalized Recommendations." Fourteenth ACM Conference on Recommender Systems. 2020.

Specific Related Works

3. CGC/PLE

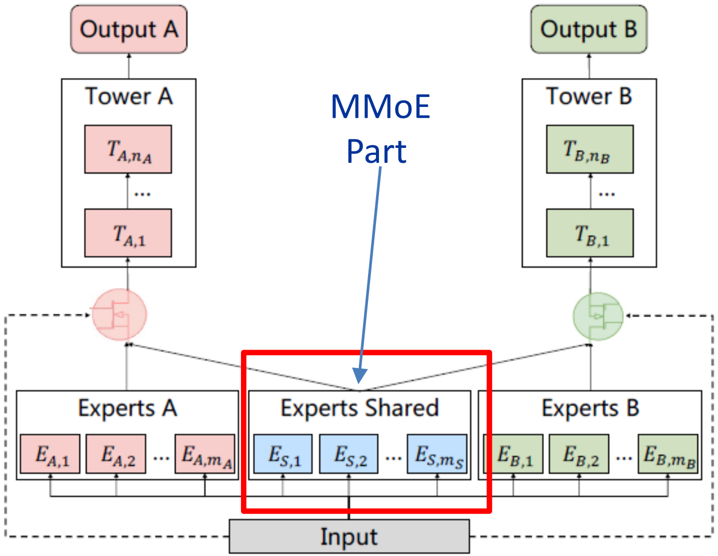


Figure 4: Customized Gate Control (CGC) Model

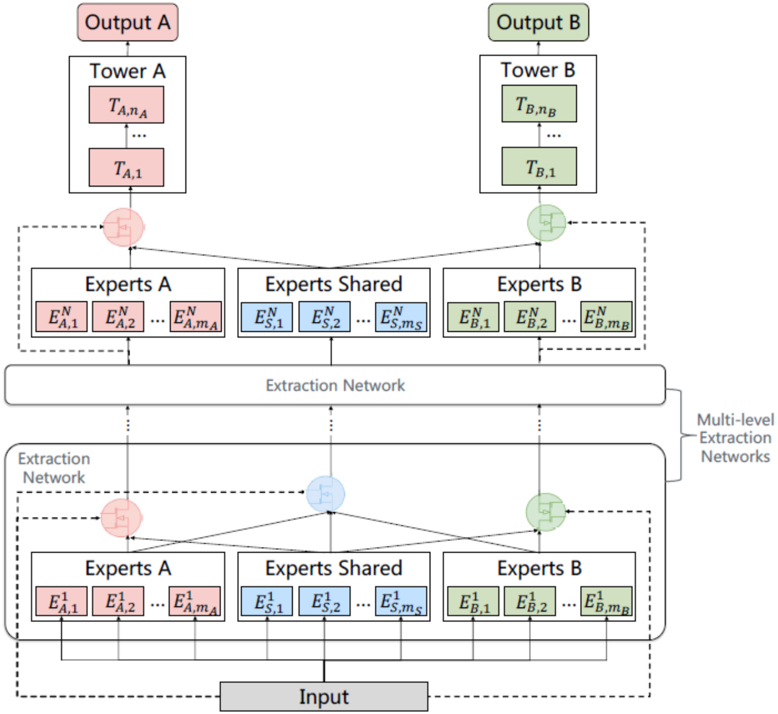


Figure 5: Progressive Layered Extraction (PLE) Model

Implementation details on Youtube-8m

1. The Baseline should be a Fully Connected Network with an output size of 3862.
2. The number of labels of youtube-8m is 3862. Do we need to build a tower layer for each category?
    a. Instead of building 3862 tower layers, the implementation in SNR is: building one tower layers for each topic(25 tower: 24 topic and 1 unknown). The output units of each topic is the number of labels in this topic where one units is corresponding to a label.

THANKS!