ML hw3 report

b02902080 資工四 郭傳駿

Supervised Learning

- convolutional neural network built with Keras and Tensorflow

- **NETWORK LAYERS**
  - conv_1 (64 , 3 , 3), activation: elu (exponential linear unit)
  - conv_2 (64 , 3 , 3), activation: elu
  - batchnormalization
  - maxpooling (3 , 3), stride(2 , 2)
  - dropout(0.25)
  - conv_3 (128 , 3 , 3), activation: elu (exponential linear unit)
  - conv_4 (128 , 3 , 3), activation: elu
  - batchnormalization
  - maxpooling (3 , 3), stride(2 , 2)
  - dropout(0.25)
  - conv_5 (256 , 3 , 3), activation: elu (exponential linear unit)
  - conv_6 (256 , 3 , 3), activation: elu
  - batchnormalization
  - maxpooling (3 , 3), stride(2 , 2)
  - dropout(0.25)
  - dense(FC)(512), activation: elu
  - dropout(0.5)
  - dense(10)
  - softmax -> classification OUTPUT

- **TRAINING METHODS**
  - do data_augmentation before fitting the CNN model
  - categorical_crossentropy as loss function
  - adadelta as optimizer
  - tried early_stopping with 60 epochs and no early_stopping with 45 epochs
  - batch_size set to 16

- **PERFORMANCE**
  - 76% at best on Kaggle public set

Self-Training

- reuse same network structure above in supervised learning
- based on model trained with labeled data, predict labels for unlabelled data
- if a prediction of an image has a maximum softmax probability higher than a threshold of 0.999, it is considered confident, otherwise not-confident

- add confident unlabelled data and their predicted classes to the training data of the network and train the network again
- update the set of not-confident data as unlabelled data for the next round
- perform 10 rounds of self-training, in each round:
  - early_stopping with 20 epochs
  - batch_size is set to 32
  - optimizer and loss function are the same as used in supervised learning
- best performance up to **82%** accuracy on Kaggle public set
- findings:
  - good unlabelled data is mostly added to training set within 10 rounds, the performance generally starts dropping from round 8
  - validation loss occasionally soars high due to overfitting during self-training, so its good to add early_stop to prevent model from getting worse, training often stops within 10 epochs

---

Clustering

- using the output of layer Dense(512) to represent an image
- do clustering by **constrained k-means** with properties
  - initial means set to be centroids of the 500 representations of each class
  - label of labeled data don't change in the process
- perform 10 rounds of k-means clustering, in each round:
  - assign labeled data to its class
  - get softmax probability from the CNN model and run over unlabelled data
  - predict class of unlabelled data by comparing L2 distance between 10 centroids
  - to ensure some level confidence, assigned unlabelled data to predicted class if
    1. softmax probability of the most likely class has value higher than a threshold of 0.99
    2. L2 distance between data representation and nearest centroid lower than 15
  - update new centroids from taking mean over assigned image representations of each class
- performance at 76.54% on Kaggle private set

Discussion

- the architecture of the convolutional network is not too complicated yet has quite impressive results, after a lot of tuning and trying, i think the best this network can do on self-training is about 82% accuracy.
- the performance of self-training depends a lot on the performance of the "pre-trained" model from supervised learning, and the performance of self-training reaches its peak mostly after 6 or 7 rounds.
- the performance of my clustering method is poor compared to both the others, one way i think to improve it is to design a way to detect convergence and run k-means clustering until convergence instead of a fixed number of rounds.