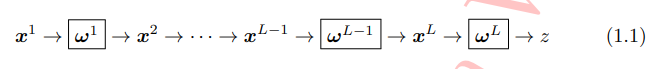
Loss function：

By calculating the loss between the predicted value and the true value, relying on the backpropagation algorithm, the loss is fed back-ward layer by layer from the last layer. The parameters of each layer are updated, and the parameters are updated again to feed forward, and so on, until the network model Convergence, so as to achieve the purpose of training the model.

Generally, Convolutional Neural Network is a hierarchical model which uses convolution and other operations as the basic units on raw data (x1 in 1.1) layer by layer, such as RGB image, raw audio data, etc ending with the calculation of the loss function and each layer of data as a 3-dimensional tensor.



If y is the ground truth corresponding to input x1, then the loss function is expressed as:



Among 1.2, the parameter in the function L(·) is ωL.

In practical applications for different tasks, the form of the loss function also changes. Taking the regression problem as an example, the commonly used ℓ2 loss function can be used as the objective function of the convolutional network1.2.1:



For classification problems, the objective function of the network often uses the cross-entropy loss function1.2.2:





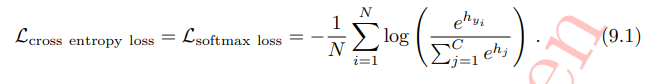
C: classification task function

There are two types of objective functions in the classical prediction tasks, which are classification and regression.

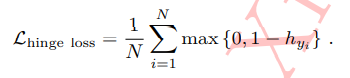
1. Objective function of classification task

For classification assignments, suppose a classification task has N training samples, the input feature for the i-th sample of the final classification layer of the network is xi, and its corresponding true label is yi ∈ {1, 2,...,C}, and h = (h1 , h2,...,hC )⊤ is the final output of the network, that is, the prediction result of sample i, where C is the number of classification tasks. The true label of the sample corresponds to a one hot vector.

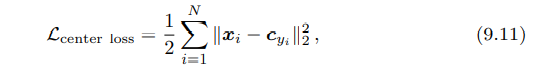
The cross entropy loss function is also called the softmax loss function.



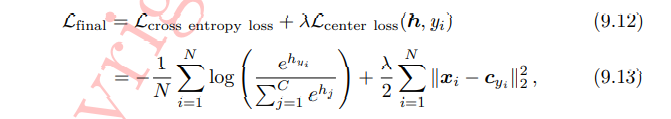
The hinge loss function is widely used in vector machines:



The central loss function also puts some attention on reducing intra-class differences while considering the distance between classes.



Where cyi is the mean ("center") of all depth features of the yi class, hence the name "central loss function". Intuitively, 9.11 forces all samples belonging to the yi class not to be too far away from their center, otherwise the penalty will be increased. In actual use, since the central loss function itself considers the difference within the class, the central loss function should be used in conjunction with other loss functions that mainly consider the distance between classes, such as the cross-entropy loss function. The final objective function of the network is as follows:



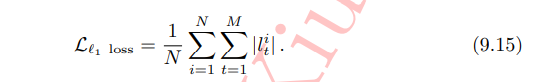
1. Objective function of regression task

The true label of the sample also corresponds to one vector, but the difference is that each dimension of the true label is a real number instead of a binary value (0 or 1). In the regression task, the result of the regression is some integers or real numbers and there is no prior probability density distribution. The commonly used losses are L1 loss and L2 loss.

The prediction error is used to measure how close the predicted value of the model is to the true label. Assuming that the true label corresponding to the i-th input feature xi in the regression problem is yi = (y1, y2,...,yM)⊤, and M is the total dimension of the label vector, then lit represents the network regression predicted value on sample i ( yˆi). The prediction error (also known as the residual) of the t-th dimension from its true label:

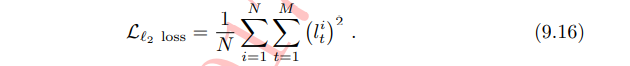


Mean absolute loss (MAE) is also called L1 Loss, which uses the absolute error as the distance:



Due to the sparseness of L1 loss, in order to penalize larger values, it is often added as a regular term to other losses as a constraint. The biggest problem of L1 loss is that the gradient is not smooth at the zero point, which causes the minimum value to be skipped.

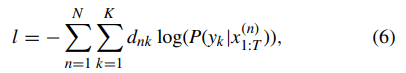
Mean Squared Loss/ Quadratic Loss (MSE loss) is also called L2 loss, or Euclidean distance, which uses the sum of squared errors as the distance:



L2 loss is also often used as a regular term. When the predicted value is very different from the target value, the gradient is easy to explode because the gradient contains x−t.

In this paper, the proposed network is asked to classify speakers using a multi-class cross entropy objective function, which is commonly used for image object detection and speaker recognition in neural networks. Previous experiments were trained to predict speaker label from frames while in the paper predicting speakers from variable-length segments. Given a dataset with N training examples from K speakers, with a speech segment consisting of T input frames . The prediction probability of the deep network model for the k-th speaker is defined as .

The cross-entropy objective function is defined as below:



where the quantity dnk is 1 if the speaker label for the n-th training segment is k; otherwise, it is 0.

Hyper parameter training:

1. Setting hyper parameter:

Before building the entire network architecture, specifying various hyperparameters related to the network structure is a must: input image pixels, number of convolution layers, convolution kernel related parameters, etc.

When using convolutional neural networks to process image problems, different input images are output with the same specifications, and at the same time, it is convenient for GPU devices to parallelize, and the images will be compressed to 2^n size uniformly.

The hyperparameters of the convolution layer mainly include the size of the convolution kernel, the step size of the convolution operation, and the number of convolution kernels.

Similar to the size of the convolution kernel, the kernel size of the convergence layer is generally set to a smaller value, which plays a role of "down sampling".

In the paper, there are several hyper parameters defined in the self-attention mechanism. The hyper-parameter nk corresponds to the number of attention hops.

1. Training:

When training a convolutional neural network, although the training data is fixed, due to the mini-batch processing training mechanism, we can train the data set before each epoch of the model is trained Randomly shuffled to ensure that the data "sees" in the same batch of different epochs of the model is different. Such processing will not only increase the rate of model convergence, but at the same time, compared to a model trained in a fixed order, this operation will slightly improve the prediction result of the model on the test set.

Another key setting during model training is the model learning rate. An ideal learning rate will promote model convergence, while an undesirable learning rate will even directly lead to the loss of the model's direct objective function.

Regarding the optimization algorithm selection of model parameters. The stochastic gradient descent method is currently the most commonly used network training method.

Grid search:

Grid search is a model hyperparameter optimization technology, which is often used to optimize three or less hyperparameters. It is essentially an exhaustive method. For each hyperparameter, the user chooses a smaller finite set to explore. Then, the Cartesian product of these hyperparameters obtains several sets of hyperparameters. Grid search uses each group of hyperparameters to train the model, and selects the hyperparameter with the smallest validation set error as the best hyperparameter.

Drawback:

Grid search uses an exhaustive approach, and its computational complexity will increase exponentially with the size of the hyperparameters that need to be optimized. Therefore, this method is only suitable for small-scale hyperparameter optimization problems.

Improvement:

When the hyperparameter scale is large, random search will be a more efficient hyperparameter optimization method. At the same time, because the pure grid search is relatively extensive for the optimization of hyperparameters, the more fine-grained restriction and processing of the grid search strategy and scope may be useful for improving the efficiency of the grid search.

In this paper, grid search is used to determine a suite of hyper-parameters such as weight decay, the size of the embedding layer, the dropout probability, and the maximum gradient norm. All experiments in the paper are implemented by the widely used deep learning tool TensorFlow. The batch size is set as 128 and train neural networks on one NVIDIA GTX 1080 Ti GPU.

In the paper, the Adam optimization algorithm is used. The Adam optimization algorithm is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing. The attractive benefits of using Adam on non-convex optimization problems are being straightforward to implement, being computationally efficient, little memory requirements, being invariant to diagonal rescale of the gradients, and being appropriate for non-stationary objectives.

Adam is different to classical stochastic gradient descent. Stochastic gradient descent maintains a single [learning rate](https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/) (termed alpha) for all weight updates and the learning rate does not change during training. That is to say, learning rate is maintained for each network weight (parameter) and separately adapted as learning unfolds.

The parameters of the Adam optimizer are alpha (learning rate or step size, the proportion that weights are updated), beta1(the exponential decay rate for the first moment estimates), beta2(the exponential decay rate for the second-moment estimates), epsilon(a small number to prevent any division by zero in the implementation)

The parameter of the Adam optimizer is used with β1 = 0.9, β2 = 0.98, epsilon = 10−9. Then adopting the warmup process for varying the learning rate increases the learning rate linearly for the first predefined training steps and then decreases it proportionally.

To reduce sensitivity to utterance length, it is desirable to train the network on speech chunks that capture the range of duration we expect to encounter at test time (in figure 5).

