Approaching Speaker Verification

Ryan Brigden
Madhura Das

11-785: Introduction to Deep Learning // Recitation 6 // October 5, 2018

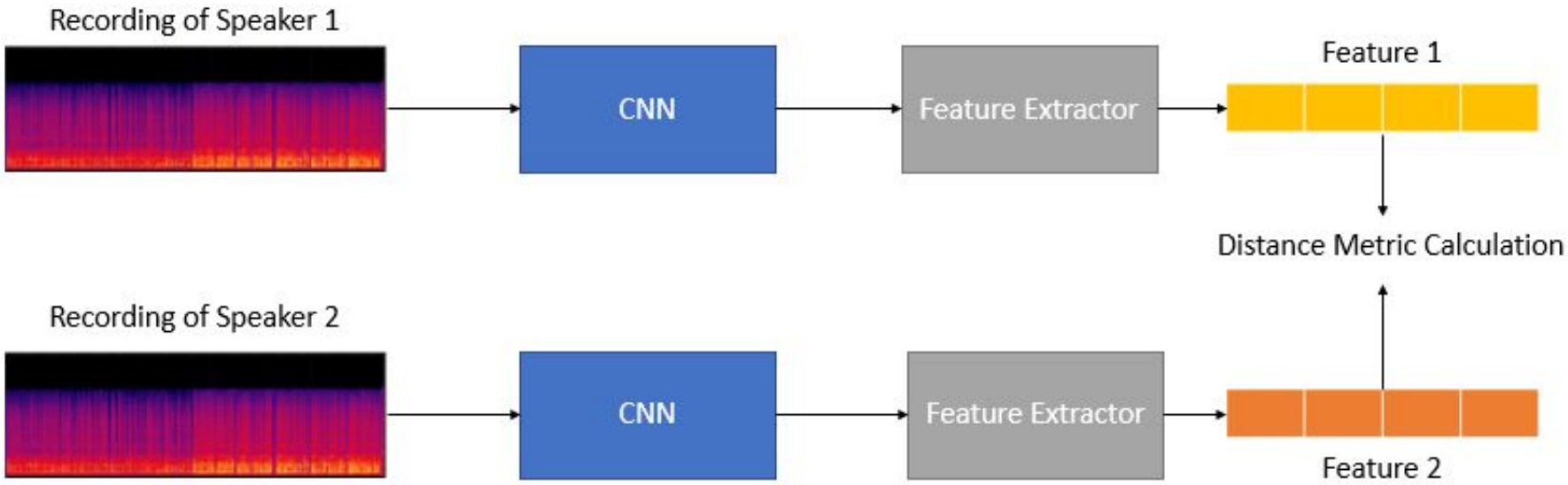
Today's Agenda

- Speaker verification
  - Verification vs identification
  - Transfer learning
  - Open set vs closed set
  - Equal error rate (EER)
- Data pre-processing and training procedure
  - Voice activity detection (VAD)
  - Handling huge datasets
  - Variable length samples
- Model Architecture
  - Experimental process
  - 2D-CNN vs 1D-CNN
  - Pooling strategies
  - ResNet, DenseNet, *Net
- Learning objective
  - Transfer learning
  - Contrastive loss
  - Margin problem
  - Center loss
  - Triplet loss
  - Softmax (revisited)
  - Angular softmax

Problem Statement

- Use transfer learning to design a system for Speaker Verification
- Speech utterances contain a lot of acoustic features that vary across different speakers
- The main task is to train a CNN to extract and represent such important features from an utterance.
- The extracted features will be represented in a fixed-length vector of features, known as a speaker embedding
- Given a pair of embeddings, we need to identify if they belong to the same speaker

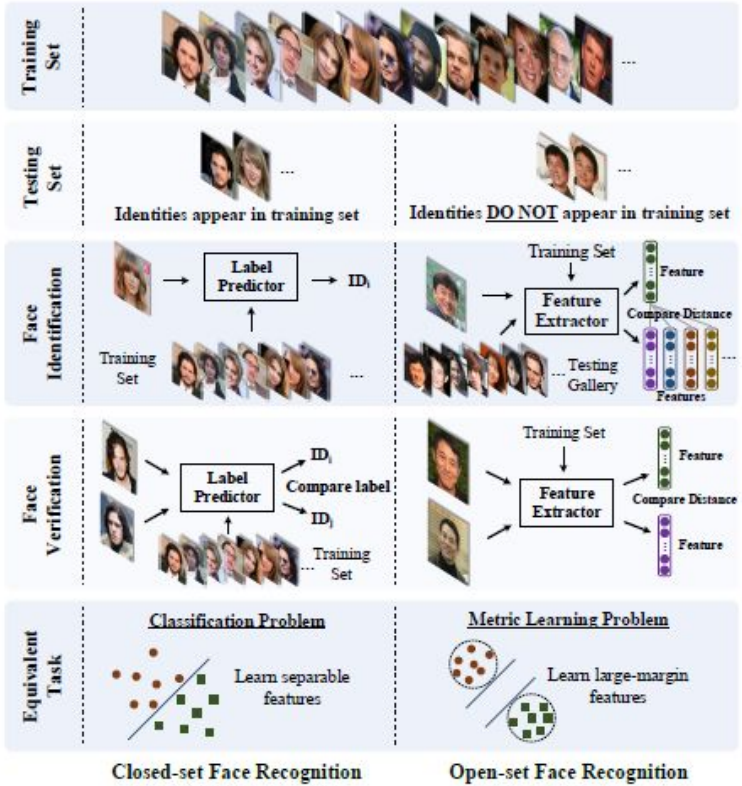
General Approach



Open Set vs Closed Set

- Closed set
  - Treated as a classification problem
  - Training dataset is assumed as exhaustive
  - All possible data classes are present in the training set
  - Identification is among previously defined classes
- Open Set
  - Learn features instead of data classes
  - Training dataset is not exhaustive
  - Used as a metric learning problem

Open Set vs Closed Set



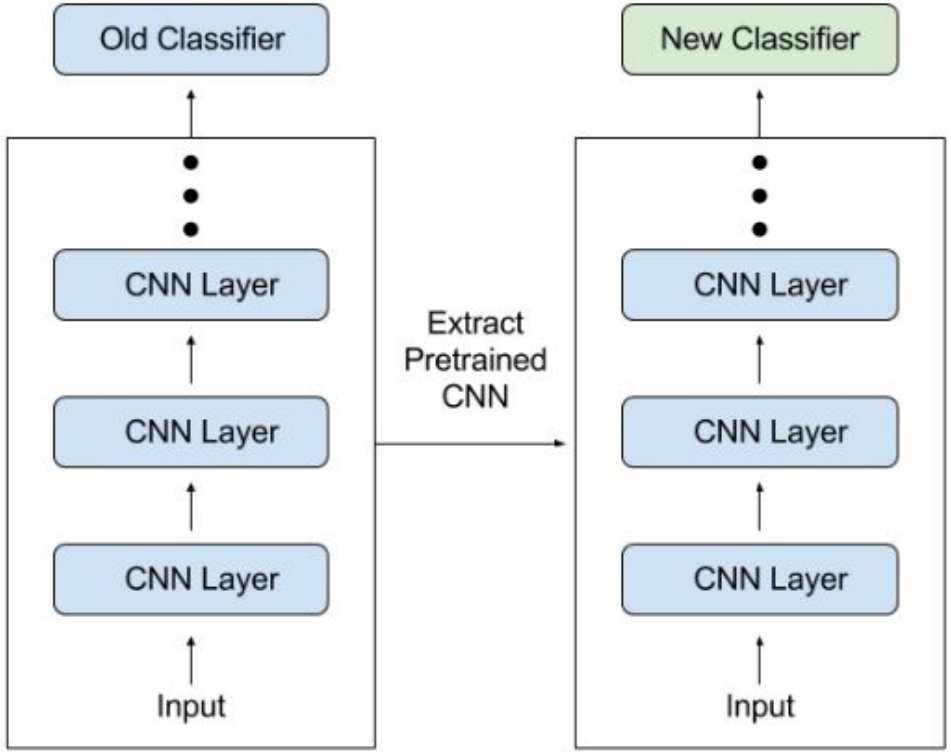
Verification versus identification and open versus closed set for the facial recognition problem

Identification vs Verification

- Identification
  - Identify/Classify the input data to the network
  - HW1 Part 2 was an identification task where we were identifying phonemes
- Verification
  - It’s a 1-to-n matching where given a sample, you match it to the closest sample among n other samples
  - Can also be performed as a 1-to-1 task where we verify if they belong to the same class of data or not
  - In this homework, your main objective is to verify

Transfer Learning

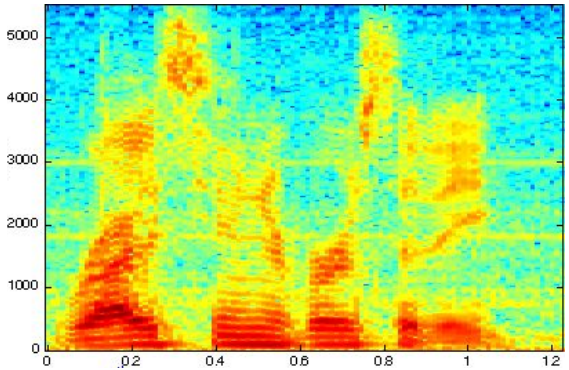
- Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task
- Develop Model Approach
  - Select a source task
  - Develop source model
  - Reuse model
  - Tune model
- Pre-trained Model Approach
  - Select source model (usually pre-trained)
  - Reuse model
  - Tune model



For the Homework...

- Transfer Learning to be used for the task of speaker verification
- We follow Develop Model Approach
- Train a convolutional network to perform N-way speaker identification
- Extract intermediate representations of speaker recordings
- Verify if two intermediate samples belong to same speaker
- Fine-tune the network to learn more discriminative features

Speaker classification



Speech sample (log-spectrogram)

CNN

Embedding

Classification

$\mathrm{softmax}(\mathbf{z})$

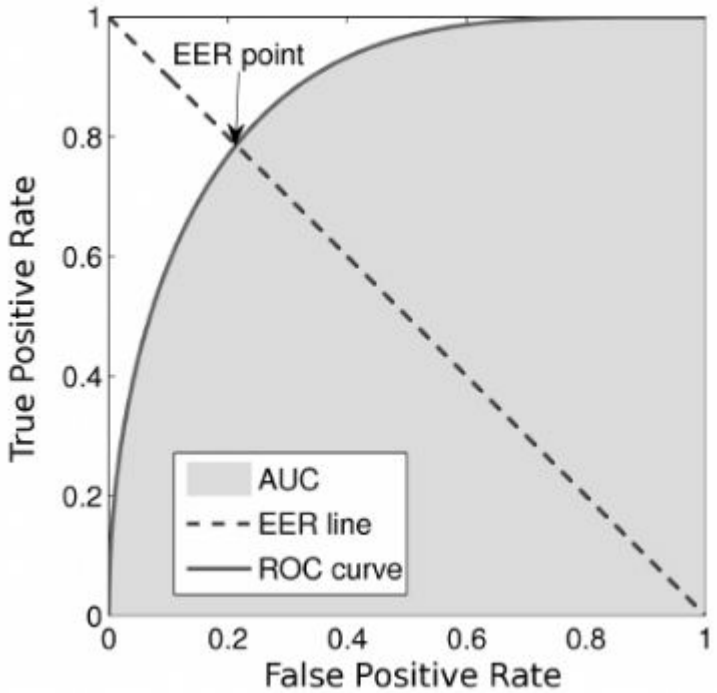
N-class probabilities

Fully-connected layers

➔ N-way classification where is N is the number of speakers present in the training set
- Cross-Entropy objective
- Applied per sample

Equal Error Rate (EER)

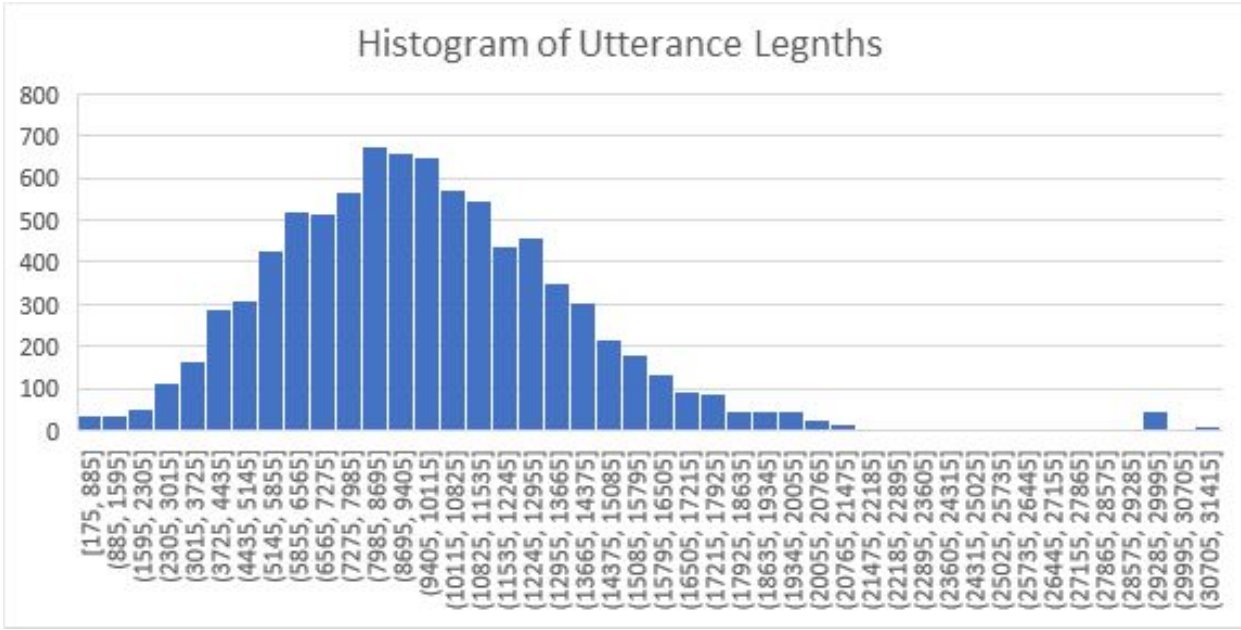
- Equal error rate (EER) is for binary classification problems used to predetermine the threshold values for its false acceptance rate and its false rejection rate
- When the rates are equal, the common value is referred to as the equal error rate
- The value indicates that the proportion of false acceptances is equal to the proportion of false rejections
- The threshold can be adjusted to meet the requirements for a specified task, but EER is a standard metric for performance.



Source: https://www.webopedia.com/TERM/E/equal_error_rate.html

Dataset Statistics

- Average recording length: 9717
- Max recording length: 30994
- Min recording length: 175
- Number of speakers
  - Chunk 1: 127
  - Chunk 1-2: 381
  - Chunk 1-3: 889



Data Processing (VAD)

- Voice Activity Detection in which the presence or absence of human speech is detected
- There may first be a noise reduction stage, e.g. via spectral subtraction
- Then some features or quantities are calculated from a section of the input signal
- A classification rule is applied to classify the section as speech or non-speech – often this classification rule finds when a value exceeds a threshold
- Helps reduce the dataset size as we remove the silence

Handling Variable Lengths

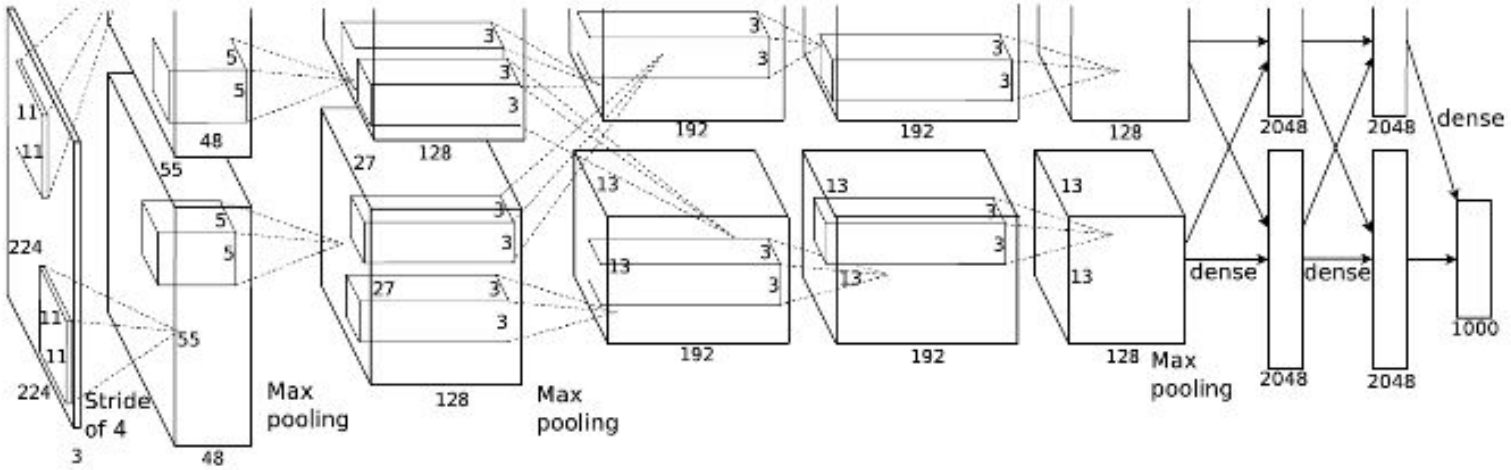
- Slice off a fixed length from each utterance
- Slice off a fixed length randomly in an utterance
- If the utterance length is lesser than your fixed length, repeat the utterance to the fixed length
- Wrap all data to a fixed length and slice off fixed lengths of data from a random point (Hint: Use np.pad)

CNNs in PyTorch

- Conv1d
  - Set number of channels as 64 (feature size)
  - While average pooling, average the feature across time dimension
  - Can achieve fairly low EERs of around 24 in first few epochs
- Conv2d
  - Treat each utterance as monochrome 2D image
  - Set number of channels to 1
  - While average pooling, average features across time dimension and then features
  - Should achieve higher EERs than a Conv1d network
- Tapering
  - Memory Optimization Technique: Layers near the input can have higher filter sizes and larger strides
  - Reduce filter sizes and strides as the network goes deeper
  - Make sure you handle your padding

CNN Architectures: AlexNet

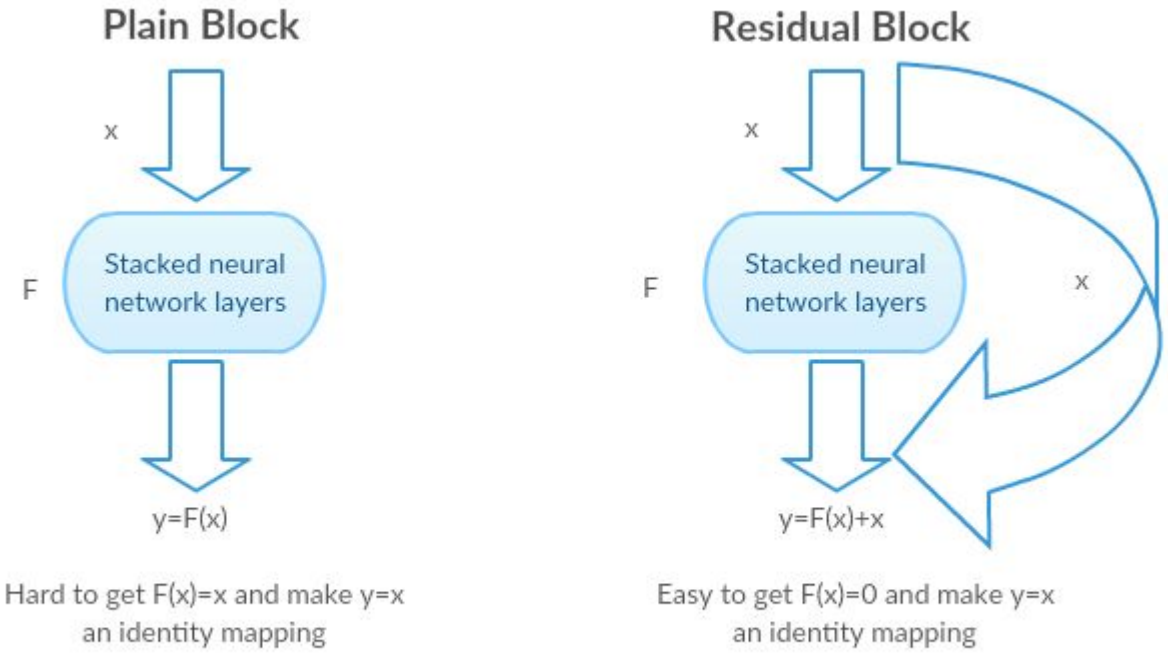
- Won the 2012 ImageNet LSVRC-2012
- Used ReLU, dropout



VGGNet

- Convolutions layers (used only 3*3 size)
- Max pooling layers (used only 2*2 size)
- Fully connected layers at end
- Total 16 layers

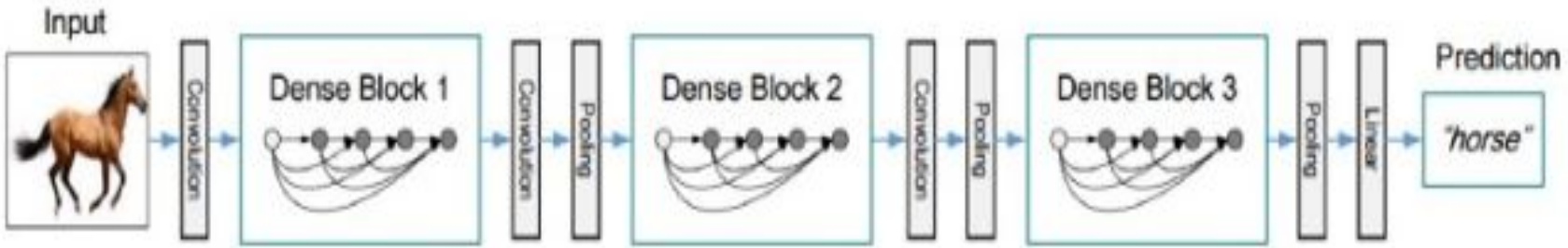
ResNets



Identity mapping in Residual blocks

DenseNets

- Extension of ResNets
- Strengthen feature propagation, encourage feature reuse
- Fewer parameters than a ResNet



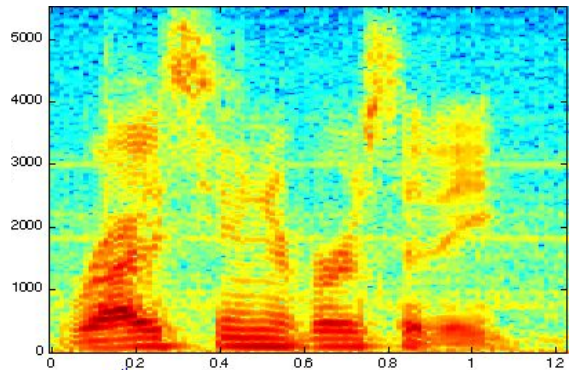
Recap

- ➔ The speaker verification problem
- ➔ Classification for verification
- ➔ Practical data considerations
- ➔ CNNs for feature learning
- ➔ Architecture variations

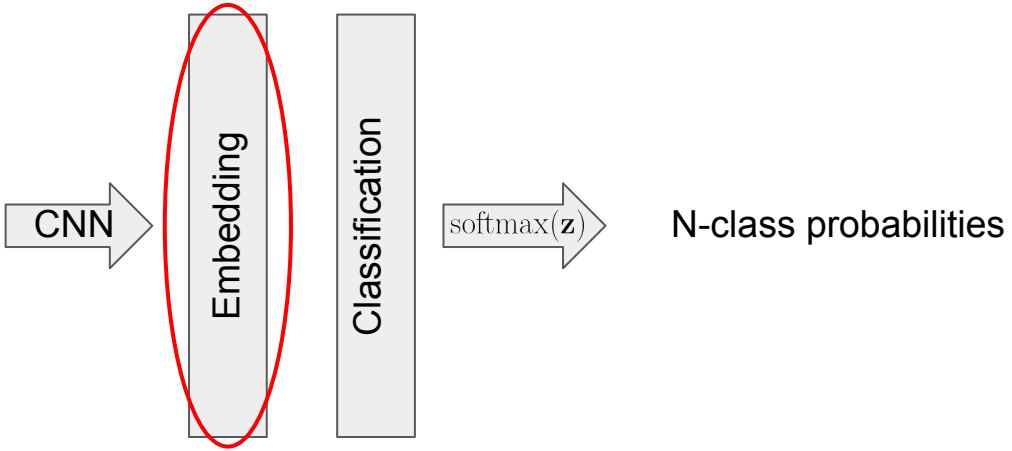
What's Next

- ➔ Classification for verification, revisited
- ➔ Explicitly optimizing for discriminative features
- ➔ Thinking about margins
- ➔ Triplet, center, contrastive losses
- ➔ Angular softmax

Speaker classification for verification



Speech sample (log-spectrogram)



Fully-connected layers

- ➔ Utilize the activations of the second to last fully-connected layer as a feature embedding.
- ➔ Estimate the similarity between two speech samples as the inverse distance between their embeddings.
- ➔ Consider the implications of different distance metrics (e.g. euclidean, cosine)

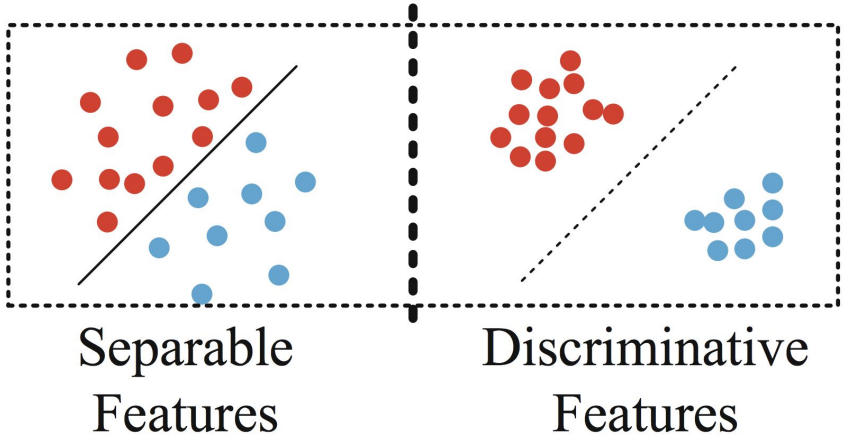
DNNs *can* learn meaningful features

Eight ILSVRC-2010 test images and the five labels considered most probable by our model. The correct label is written under each image, and the probability assigned to the correct label is also shown with a red bar (if it happens to be in the top 5).

Five ILSVRC-2010 test images in the first column. The remaining columns show the six training images that produce feature vectors in the last hidden layer with the smallest Euclidean distance from the feature vector for the test image.

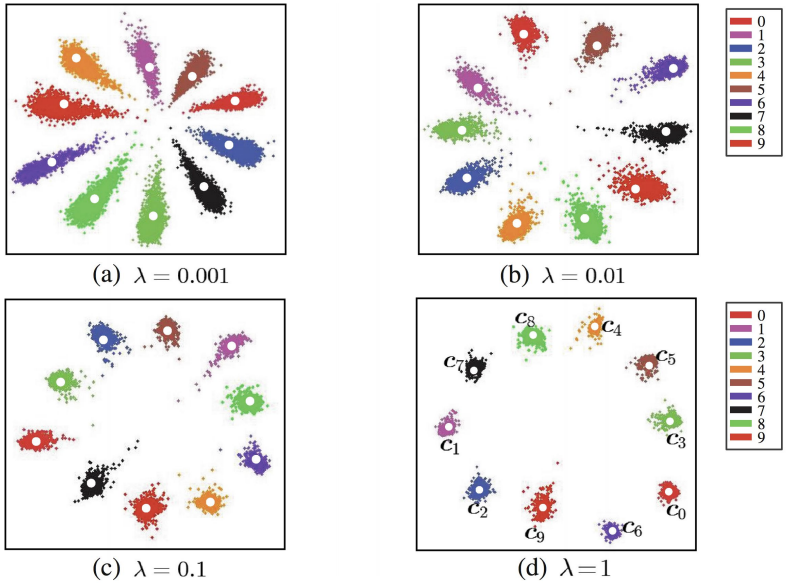
Discriminative features

- ➔ Classification task optimizes learning separable features
- ➔ Optimally we wish to learn discriminative features
  - ◆ **Maximal** *inter* class distance
  - ◆ **Minimal** *intra* class distance
- ➔ First define a measure of distance
- ➔ Most often consider Euclidean distance



Center loss

- ➔ Define a criterion that promotes minimal intra-class distance while maintaining inter-class separability
- ➔ Does **not** explicitly try to maximize inter-class class distance



(a) $\lambda = 0.001$ (b) $\lambda = 0.01$ (c) $\lambda = 0.1$ (d) $\lambda = 1$

Center Loss:

$$\mathcal{L}_c = \frac{1}{2}\sum_{i=1}^{m} \left\| x_i - c_{y_i} \right\|_2^2$$

Joint Objective:

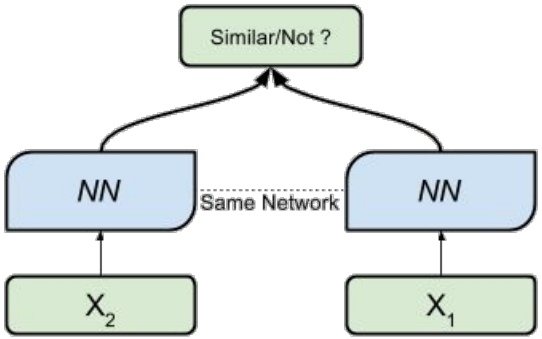
$$\mathcal{L} = \mathcal{L}_s + \lambda \mathcal{L}_c$$

c_{y_i} feature center for class label

$\mathcal{L}_s$ softmax loss (softmax + xent)

Contrastive loss

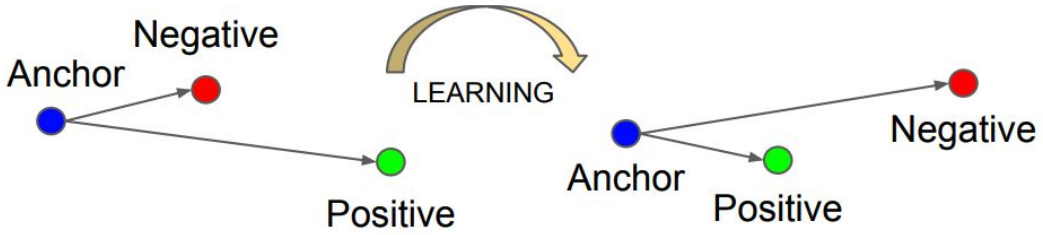
- ➔ For every pair of training instances, try to push the features for intra-class samples together and maintain some margin *m* between inter-class ones.
- ➔ Feedforward two samples *i, j* and apply the contrastive loss function provided some distance metric of choice *D.*
- ➔ Generally only useful for fine tuning. Not an information rich criterion
  - ◆ 1 bit of information per sample pair
- ➔ Tricky to balance sample pairs and choose the margin



$$\mathcal{L}_{CT}(\mathbf{z}_i, \mathbf{z}_j) = 1(y_i = y_j)\left(\frac{1}{2}D(\mathbf{z}_i, \mathbf{z}_j)^2\right) + 1(y_i \neq y_j)\frac{1}{2}(\max(0, m - D(\mathbf{z}_i, \mathbf{z}_j)^2))$$

y_k class label of sample *k* $\mathbf{z}_k$ embedding of sample *k* m margin

Triplet loss



- ➔ Motivated by nearest neighbor classification
- ➔ Try to ensure that the speaker embedding of a speaker *i* is closer to all the embeddings of that same speaker than it is to any other embedding that belongs to a speaker *i != j*
- ➔ Share parameters among 3 networks (similar to contrastive loss scenario)
- ➔ Good results when sampling triplets in an intelligent manner (sample mining)

Softmax loss (revisited)

$$p_1 = \frac{\exp(\mathbf{W}_1^T x + b_1)}{\exp(\mathbf{W}_1^T x + b_1) + \exp(\mathbf{W}_2^T x + b_2)}$$

$$p_2 = \frac{\exp(\mathbf{W}_2^T x + b_2)}{\exp(\mathbf{W}_1^T x + b_1) + \exp(\mathbf{W}_2^T x + b_2)}$$

$$(W_1 - W_2)x + b_1 - b_2 = 0$$

Resulting decision boundary (binary case)

➔ Define softmax *loss* as a combination of the final layer parameters, the softmax operator and cross-entropy loss.

➔ For this example, assume two classes (binary classification).

Angular softmax loss

- ➔ Idea: Combine Euclidean margin constraints with the softmax loss
- ➔ Formulate an angular interpretation to softmax that can be projected onto a hypersheric manifold
- ➔ No need for sample mining when identifying pairs / triplets for contrastive / triplet losses
- ➔ Interpretable as learning features that are discriminative

Angular softmax loss

Recall binary classification with softmax decision boundary:

$$(W_1 - W_2)x + b_1 - b_2 = 0$$

Angular softmax loss

Recall binary classification with softmax decision boundary:

$$(W_1 - W_2)x + b_1 - b_2 = 0$$

Note that,

$$W_i^T x_i + b_i = \|W_i^T\| \, \|x_i\| \cos(\theta_i)$$

Where θ_i is the angle between the weight and feature vector.

Angular softmax loss

Recall binary classification with softmax decision boundary:

$$(W_1 - W_2)x + b_1 - b_2 = 0$$

Note that,

$$W_i^T x_i + b_i = \|W_i^T\| \, \|x_i\| \cos(\theta_i)$$

If $(||W_i|| = 1, b_i = 0)$, we can express the decision boundary equivalently as,

$$\cos(\theta_1) - \cos(\theta_2) = 0$$

Angular softmax loss

Recall binary classification with softmax decision boundary:

$$(W_1 - W_2)x + b_1 - b_2 = 0$$

Note that,

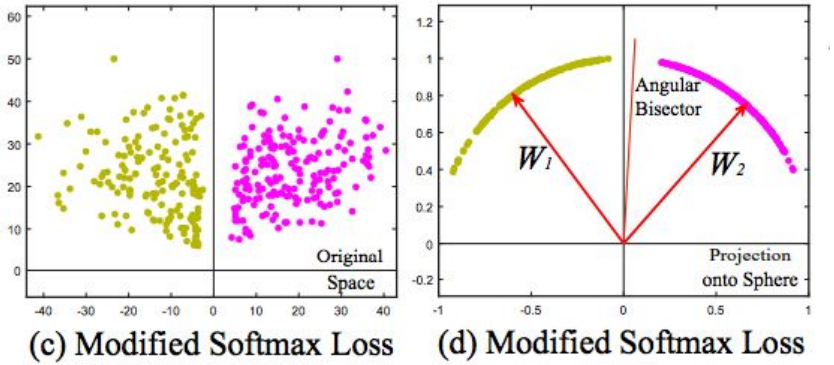
$$W_i^T x_i + b_i = \|W_i^T\| \, \|x_i\| \cos(\theta_i)$$

If $(||W_i|| = 1, b_i = 0)$, we can express the decision boundary equivalently as,

$$\cos(\theta_1) - \cos(\theta_2) = 0$$

This is simply the angle bisector of the two parameter vectors

Angular softmax loss



(c) Modified Softmax Loss (d) Modified Softmax Loss

Generalize to multi-class:

$$L_{\text{modified}} = \frac{1}{N}\sum_i -\log\left(\frac{e^{\|\boldsymbol{x}_i\|\cos(\theta_{y_i,i})}}{\sum_j e^{\|\boldsymbol{x}_i\|\cos(\theta_{j,i})}}\right)$$

$$\theta_{j,i}\ (0 \le \theta_{j,i} \le \pi),\ \text{for}\ W_j, x_i$$

Angular softmax loss

Add an angular margin *m*

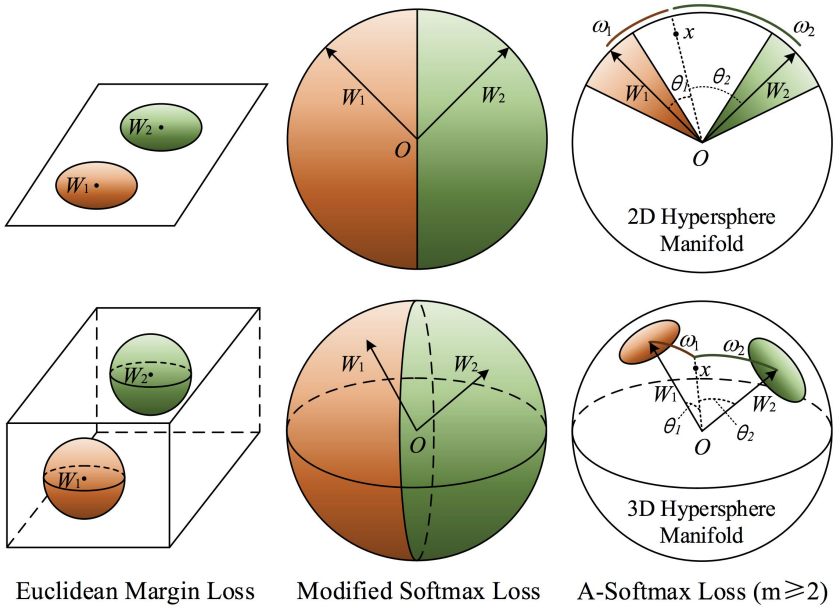
$$L_{\text{ang}} = \frac{1}{N}\sum_i -\log\left(\frac{e^{\|\boldsymbol{x}_i\|\cos(m\theta_{y_i,i})}}{e^{\|\boldsymbol{x}_i\|\cos(m\theta_{y_i,i})} + \sum_{j\neq y_i} e^{\|\boldsymbol{x}_i\|\cos(\theta_{j,i})}}\right)$$

In the binary case,

To classify type 1 correctly $\theta_1 < \frac{\theta_2}{m}$

To classify type 2 correctly $\theta_2 < \frac{\theta_1}{m}$

Angular softmax loss



➔ Nice geometric interpretation when weights are normalized and biases set to zero (in the last fully-connected layer)

Sources

- https://arxiv.org/pdf/1512.03385.pdf
- https://medium.com/@14prakash/understanding-and-implementing-architectures-of-resnet-and-resnext-for-state-of-the-art-image-cf51669e1624
- https://towardsdatascience.com/densenet-2810936aeebb
- https://arxiv.org/pdf/1608.06993v3.pdf
- https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
- https://arxiv.org/pdf/1409.1556.pdf
- https://www.ece.rice.edu/~dhj/courses/elec241/spectrogram.html
- https://arxiv.org/pdf/1704.08063.pdf
- http://ydwen.github.io/papers/WenECCV16.pdf
- https://arxiv.org/pdf/1503.03832v3.pdf
- http://yann.lecun.com/exdb/publis/pdf/hadsell-chopra-lecun-06.pdf
- https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf