

Building a Speaker Identification System from Scratch with Deep Learning



Oscar Knagg · [Follow](#)

Published in Analytics Vidhya

10 min read · Oct 2, 2018



Source: freepik.com

Neural networks are the state of the art in many classification problems, particularly those on perceptual data such as images, video, and audio. One such classification problem is to determine the identity of a speaker from an audio

sample of their voice. A model that can perform this well has applications ranging from biometric authentication to enhancing text-to-speech captioning with speaker identity. *In this post, I will outline how I made a proof-of-concept speaker identification system using convolutional neural networks trained on public source raw audio data.*

This blog post is broken into the following parts:

- Introduction: I frame the problem of few-shot learning and why it is necessary for a speaker identification system, and introduce siamese networks, an architecture designed to perform well at the few-shot learning.
- Method: Details about the dataset, training regime and some tuning experiments.
- Results: Performance of the final model and a visualization of the embedding space that it has learned.

Introduction

Conventional wisdom dictates that neural networks take huge amounts training data to reach high performance. Indeed that appears to be the case when training networks from scratch, state of the art image recognition models are typically trained on ImageNet, a dataset of 1.2 million images painstakingly labelled into 1000 classes. It's definitely possible to train an image classifier for many common objects if enough effort is spent on data collection, however obtaining 1000's of images for every conceivable variation of object is clearly impossible. The same problem holds for a speaker identification system — we simply cannot collect large amounts of labelled audio from every person in the world.

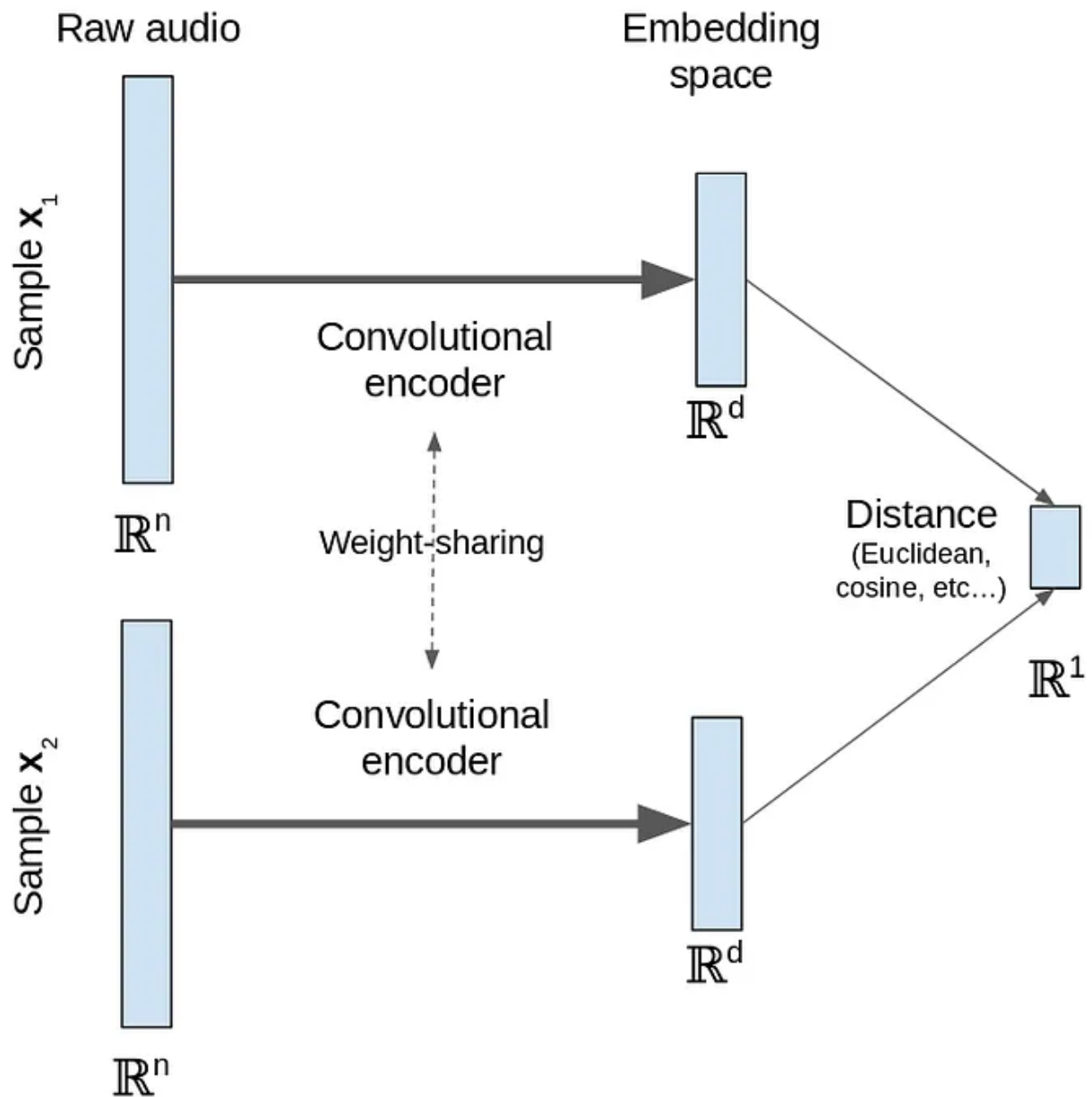
Unlike neural nets, humans are clearly capable of learning quickly from few examples. After seeing a new animal once would you be able to recognize it again afterwards? After talking to a new colleague for a few minutes would you be able to identify their voice the next day? Neural nets have a low sample efficiency compared to humans i.e. they need a lot of data to achieve good performance. Developing ways to improve the sample efficiency of learning algorithms is an area of active research and there are already promising approaches including transfer learning and meta learning among others.

These approaches are based on the intuition that learning a new task/classification problem should be easier once we have already learnt to perform many previous tasks/classification problems as we can leverage previous knowledge. In fact only *untrained* neural nets have such an absurdly low sample efficiency as it takes a lot of data to shape an untrained neural network (which is just a collection of arrays filled with random numbers) into something useful.

During training, image classifiers learn a hierarchy of useful features: first edge and colour detectors, then more complex shapes and textures. Finally, neurons deep in a network may have a correspondence to quite high level visual concepts (there is a wealth of work exploring this, including [this](#) brilliant publication). As an already trained neural network is already able to generate useful, discriminative features, it stands to reason that one should be able to leverage this to quickly adapt to previously unseen classes.

Siamese Networks

This project hinges on the use of Siamese neural networks. Unlike most common neural network architectures these networks take two separate samples as inputs instead of just one. Each of these two samples is mapped from a high-dimensional input space into a low-dimensional space by an encoder network. The “siamese” nomenclature comes from the fact that the two encoder networks are “twins” as they share the same weights and learn the same function.



Schematic of a siamese network. The samples are mapped from a high-dimensional space to a low dimensional space i.e. $n \gg d$ and then a distance measure is calculated between them.

These two networks are then joined at the top by a layer that calculates a measure of distance (e.g. euclidean distance) between the two samples in the embedding space. The network is trained to make this distance small for similar samples and large for dissimilar samples. I leave the definition of similar and dissimilar open here but typically this is based on whether the samples are from the same class in a labelled dataset.

Hence when we train the siamese network it is learning to map samples from the input space (raw audio in this case) into a low-dimensional embedding space that is easier to work with. By including this distance layer we are trying to optimize

the properties of the embedding directly instead of optimizing for classification accuracy.

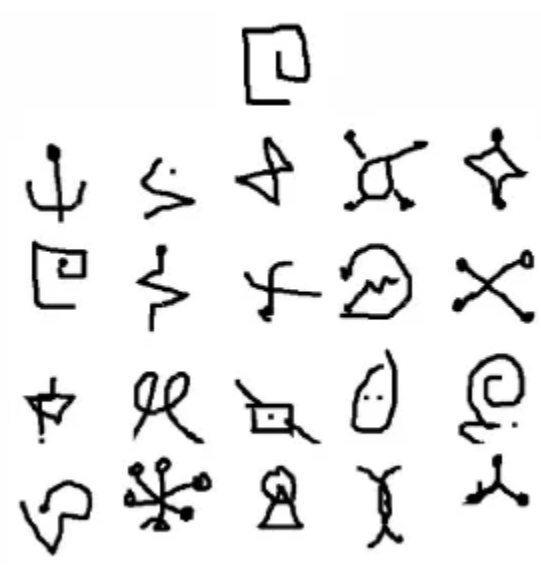
Siamese networks were introduced in 1994 by [Bromley et al.](#) (including LeCun) to verify matches between handwritten signatures. However they were repurposed by [Koch et al.](#) in 2015 for one-shot learning and it is this paper that inspired my work.

n-shot Learning

The ability to quickly generalize to previously unseen classes is often framed as the ability to perform **n-shot learning** i.e. the ability to recognize a previously unseen class after having only seen n (where n is small) examples. A model's performance at few-shot learning is measured with n -shot, k -way classification tasks which are run as follows:

1. A model is given a query sample belonging to a new, unseen class
2. It is also given a support set consisting of n examples each from k different unseen classes
3. The model then has to identify which sample(s) in the support belong to the class of the query sample

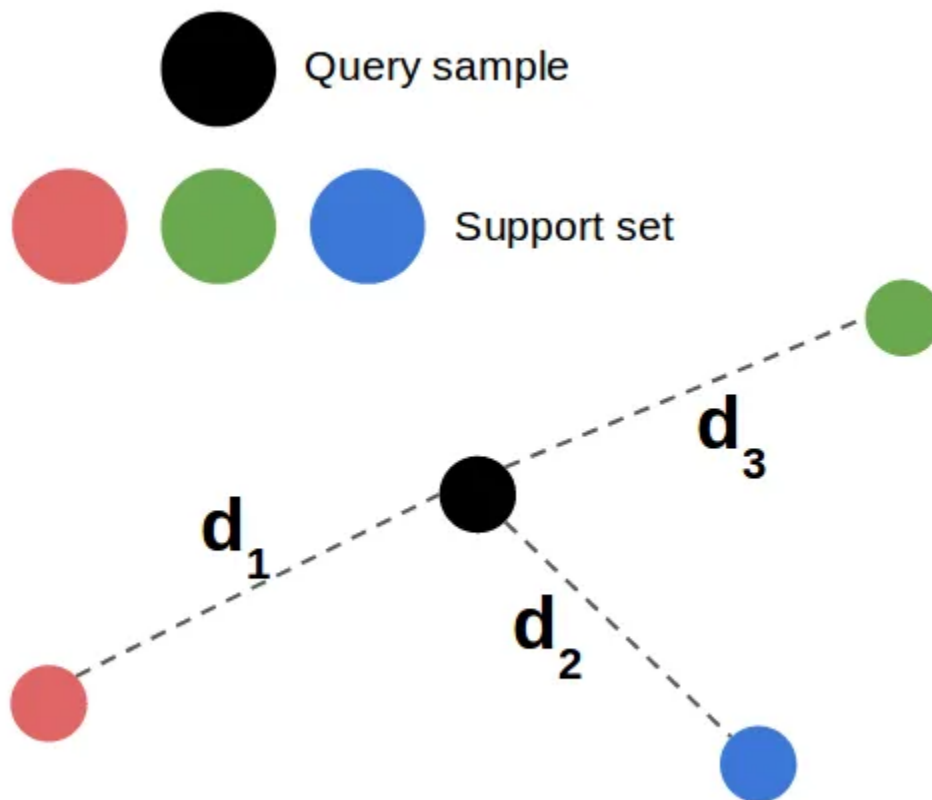
Personally I think that calling this an n -shot classification task is somewhat misleading as unlike in classification the model does not have to distinguish between all new and previously learnt classes. Instead it is more of a matching problem on previously unseen classes.



20-way 1-shot classification on the Omniglot dataset. Each of the 20 symbols is the first instance of its class seen by the model. Reproduced from [Koch et al.](#)

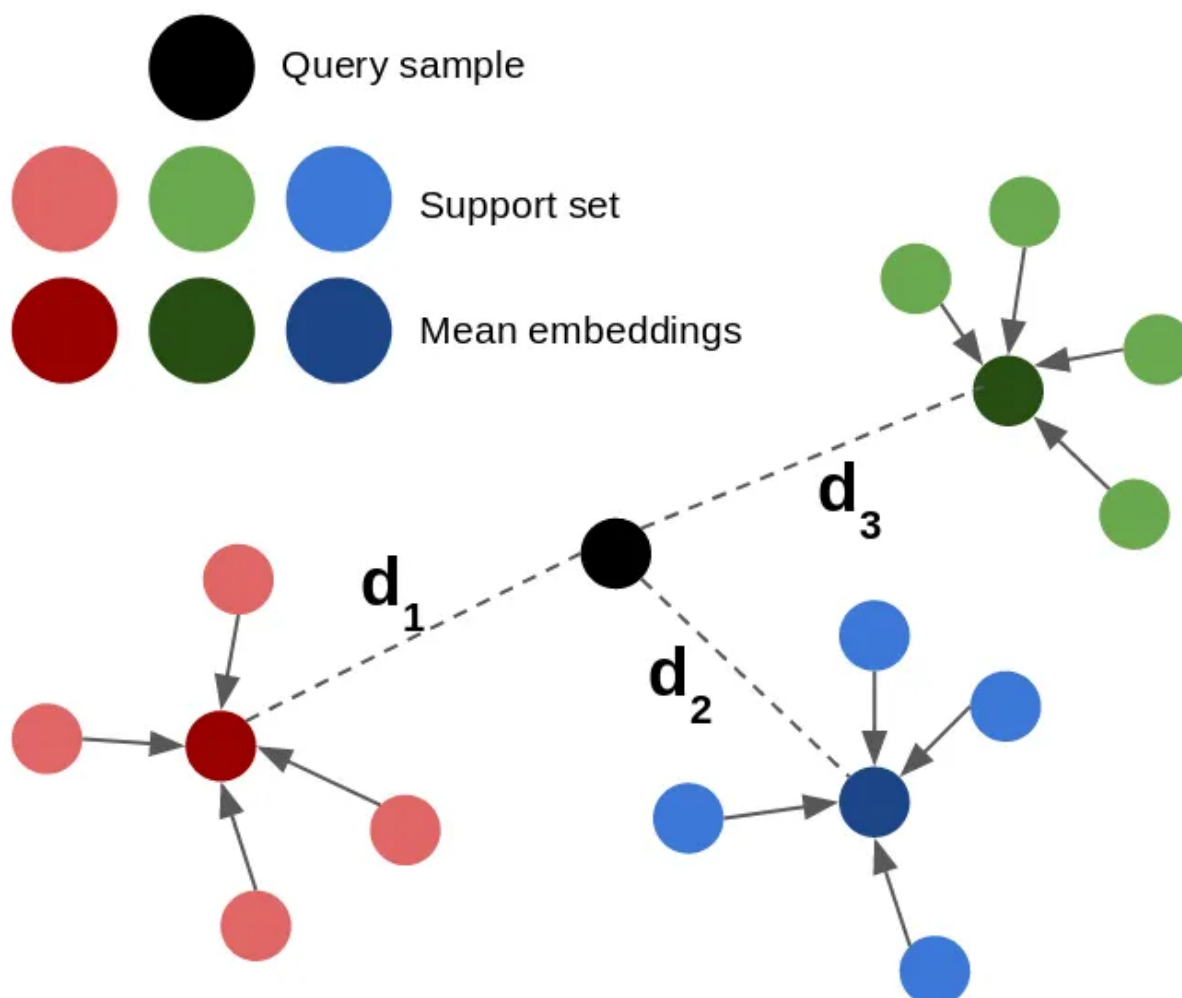
Siamese networks for n-shot learning

I expect an idea of how one could use the embedding space and pairwise similarity metric of a siamese network to perform one-shot classification is forming in your minds. The procedure is to calculate the pairwise similarity metric between the query sample and all of the support set samples by feeding them as inputs to the siamese network. Then pick the support set sample with the greatest similarity — simple!



1-shot classification. The prediction of the model is the class of the sample with the minimum distance (d_1 , d_2 , d_3) to the query sample.

One way to extend this to n -shot tasks is to convert the query samples and all of the support set samples into their representation in the embedding space and then perform a nearest-neighbour classification. I choose a slightly different approach (following [Snell et al](#)) and first calculate the mean position of the embeddings belonging to each class, then take the mean embedding that is closest to the query sample embedding to be the best class.



n-shot classification with $n=4$. The prediction of the model is the class with the minimum distance (d_1 , d_2 , d_3) from its mean embedding to the query sample.

Another approach to n-shot learning would be to use the penultimate, “bottleneck”, layer of a traditional classifier as the embedding space. However the hypothesis is that siamese networks are better suited to generalization to previously unseen classes. This is because the encoder sub-networks are trained to minimize the distance between samples of the same class and maximize distance between samples of different classes in the embedding space.

The idea is that as we are explicitly learning to differentiate between samples of different, arbitrary, classes as opposed to identifying particular classes, the learnt embedding function will be better at separating previously unseen classes than the bottleneck layer of a typical classifier. I test this hypothesis later .

Implementation

Dataset

I use the LibriSpeech corpus of audiobook data to train and evaluate models. I use the `train-clean-100` and `train-clean-360` subsets (~460 hours, >1000 speakers) for training and the `dev-clean` subset (~10 hours, 40 speakers for validation. This data contains audio from controlled environments no external noise just recording artifacts such as microphone buzz. The LibriSpeech corpus is available free of charge.

In all experiments I downsample the audio from 16 KHz to 4 KHz for quicker experimentation and reduced computational load.

Model

I use a simple 4 layer convolutional encoder network throughout this work. Architecture is as follows

- 1D convolution with large (size 32) filters followed by batch normalisation and max pooling with size 4 and stride 4
- 3 times 1D convolutions of size 3 followed batch normalisation and max pooling with size 2 and stride 2
- Global max pooling and a dense layer to produce the embeddings
- The siamese network is two of the above networks (with weight sharing) joined by a euclidean distance layer
- The final layer is a dense layer with sigmoid activation

I use ReLu activation everywhere except the final layer. I choose to perform quite aggressive max-pooling in the first layer as the initial size of the samples is quite large.

Training

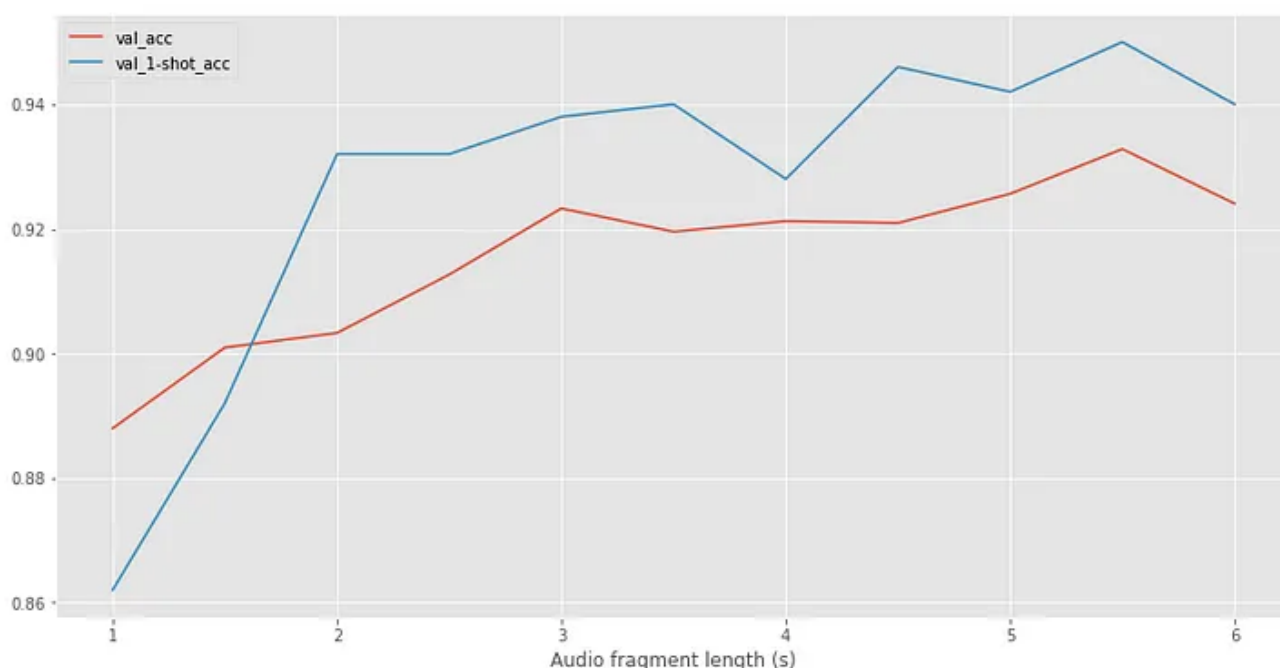
In each experiment I trained a siamese networks on batches of 32 similar and 32 dissimilar pairs for 50 “epochs” (each epoch is 1000 batches). The per-pair labels are 0 for similar pairs and 1 for dissimilar pairs i.e. pairs from same and different speakers.

I used the Adam optimizer with a learning rate of 0.001 of throughout. The 1-shot classification accuracy on a held out validation set of previously unseen speakers

was used to determine the learning rate schedule — when this metric plateaus for 10 epochs the learning rate is dropped by a factor of 10.

Audio fragment length tuning

One key parameter is the length of the audio to feed as input to the model. Intuitively one expects the accuracy of the model to increase as it is fed richer input data however the trade-off is increased training time and memory requirements.



Audio fragment length vs validation-set metrics

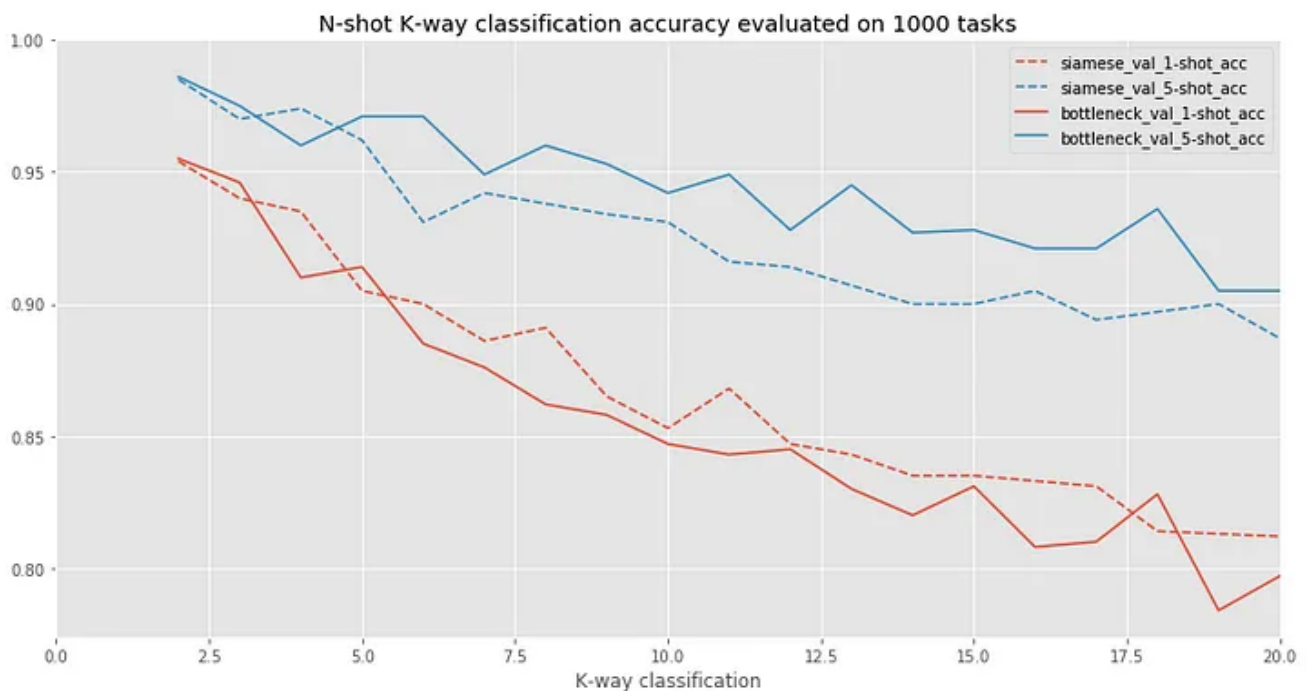
One can see that both validation 1-shot accuracy and verification accuracy keep on increasing with audio fragment length. I selected a length of 3 seconds as there appears to be diminishing returns after this point.

Hyperparameter grid search

I performed a grid search on the following hyperparameter space: initial number of convolutional filters (16, 32, 64, 128), embedding dimension (32, 64, 128, 256, 512) and dropout fraction (0, 0.1). Best results were achieved with 128 filters, embedding dimension of 64 and no dropout.

Results

Below are the results of the best siamese network on 1-shot, k-way and 5-shot, k-way classification tasks for $2 \leq k \leq 20$. I also trained a classifier network on the same dataset, using the same architecture and hyperparameters as a single encoder “twin” with the addition of a 1172 way softmax after the bottleneck layer and a categorical crossentropy loss.



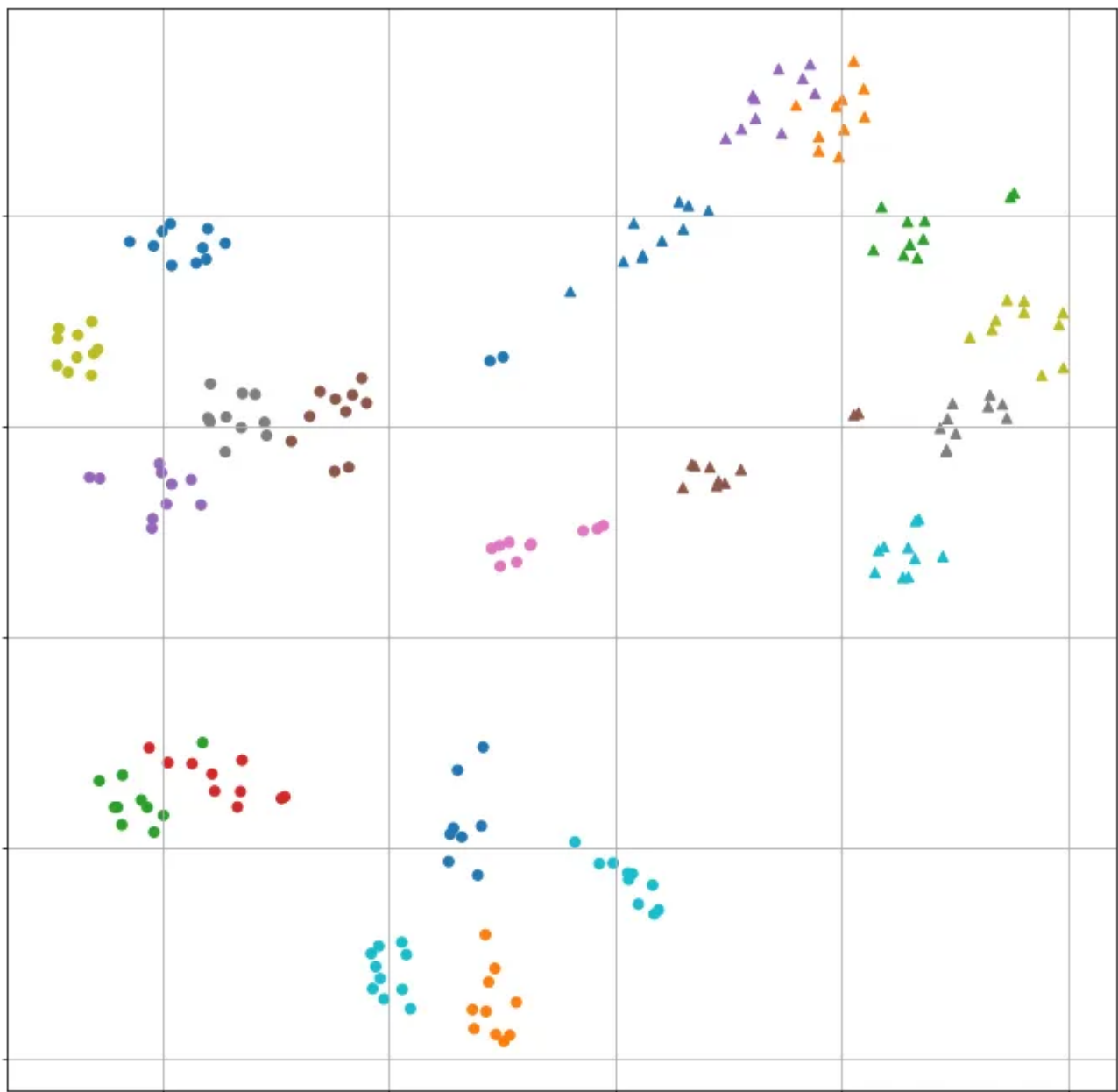
There are two results to note:

1. The siamese network does (slightly) outperform the classifier bottleneck embedding at 1-shot classification as hoped
2. Unexpectedly the classifier bottleneck embedding performs better at 5-shot classification than the siamese network with identical hyperparams

Note that the 5-shot classification performance gap between the classifier bottleneck embedding and the siamese network embedding is larger for higher k . Potentially this could be because classifier learns a better embedding for distinguishing between many classes due to the nature of the labels it uses when training. Consider that when using one-hot classification labels the softmax score of the correct speaker is pushed up and the softmax scores of all the other speakers is pushed down.

Compare this to verification tasks where training pushes the embedding distance between samples of different speakers apart, but only for the different speakers present in the batch. In particular I expect the siamese network to quickly learn to differentiate between male and female voices and that feeding it opposite-gender pairs will provide almost no learning in later epochs because they will be “too easy”. One approach to combat this would be to perform hard-negative mining to create more difficult verification tasks.

Embedding space visualisation



Circle = male, triangle = female. Each colour corresponds to a different speaker identity.

Shown above is a visualisation of the embedding space learnt by the siamese network. I selected 20 speakers from the training set and random and for each of these I selected 10 random audio samples. I then used tSNE to embed the 64-dimensional points that represent audio samples into a 2-dimensional space for plotting. Note that there is good clustering of speaker identities (as represented with colour) and a general separation of male and female speakers (as indicated by circular or triangular markers).

End Notes

In this post I've shown that it's possible to build a proof-of-concept speaker identification model that leverages recent advances in few-shot learning. I've also discovered that siamese networks are not universally superior to regular classifiers at few-shot speaker identification on the Librispeech dataset. While I have not achieved stellar performance metrics (92% verification accuracy on the validation set), but the best model is still performing enough to be used in a low security application. In further posts I intend to investigate performance improvements with the end goal of producing a pip-installable, pure-python package for speaker identification.

. . .

Code is available on my Github

<https://github.com/oscarknagg/voicemap>

Machine Learning

Speech Recognition

Keras

Audio Processing

Deep Learning



Written by Oscar Knagg

677 Followers · Writer for Analytics Vidhya

I like to build novel things

More from Oscar Knagg and Analytics Vidhya



Oscar Knagg in Towards Data Science

An intuitive guide to Gaussian processes

A maths-free explanation of an underappreciated algorithm

Jan 15, 2019



1.8K



11





Kia Eisinga in Analytics Vidhya

How to create a Python library

Ever wanted to create a Python library, albeit for your team at work or for some open source project online? In this blog you will learn...

Jan 26, 2020 2.6K 28



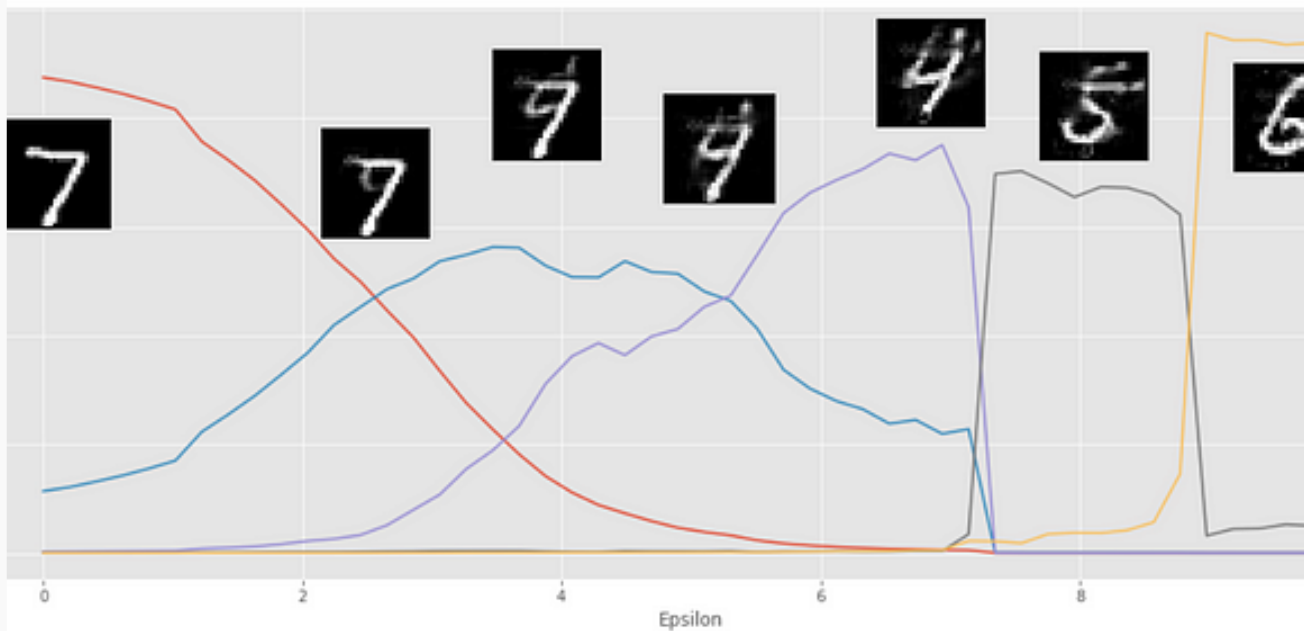
Harikrishnan N B in Analytics Vidhya


Confusion Matrix, Accuracy, Precision, Recall, F1 Score

Binary Classification Metric

Dec 10, 2019 1K 6





 Oscar Knagg in Towards Data Science

Know your enemy

How you can create and defend against adversarial attacks

Jan 6, 2019  559  4



See all from Oscar Knagg

Recommended from Medium



 Mark O'Brien

Whisper & Python for Video Transcription

Explore the Whisper Model from OpenAI for audio transcription made easy, fully local

★ Mar 4 🖱 6



 Arun Viswanathan

The Sound of Learning: Using OpenAI's Text-to-Speech API for a Simple Dictation Game for Kids

Update (March 1st 2024): The code in Github has been updated to alternately use the Google text-to-speech API as well which has much better...

★ Feb 24 🖱 14

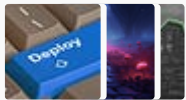


Lists



Practical Guides to Machine Learning

10 stories · 1635 saves



Predictive Modeling w/ Python

20 stories · 1357 saves



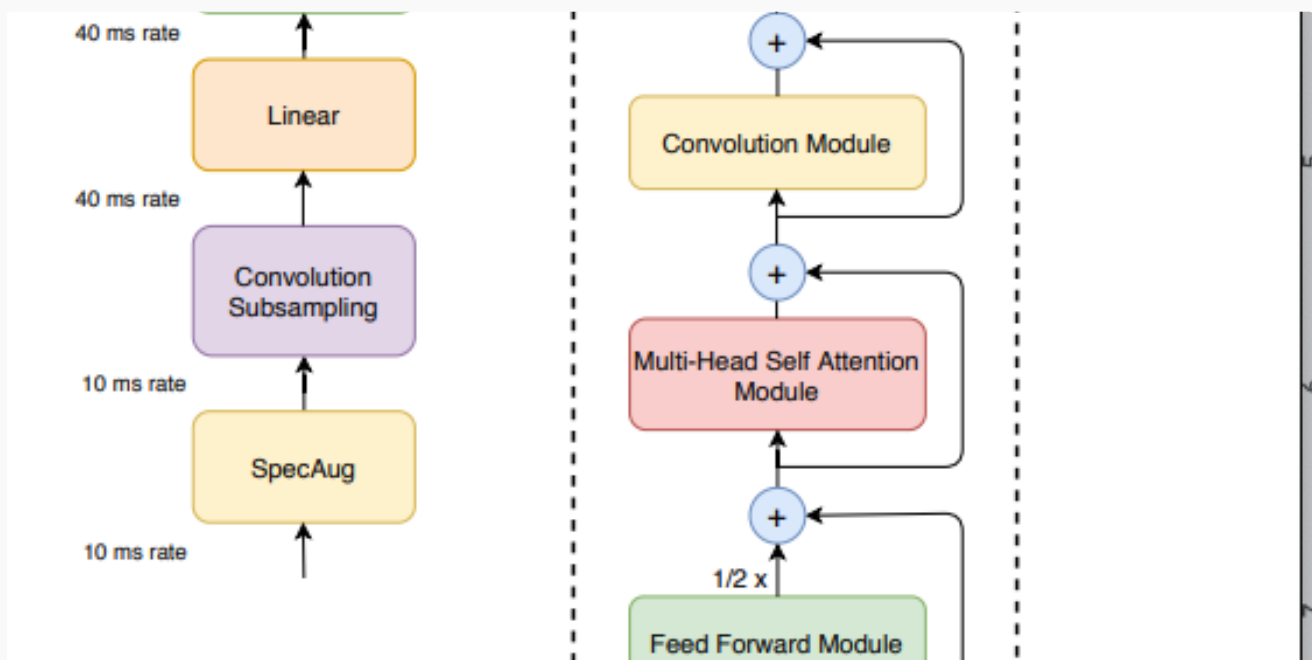
Natural Language Processing

1566 stories · 1112 saves



data science and AI

40 stories · 197 saves



Manish Chablani

ASR ML Systems: Overview and latest model architectures: Transducers, TDT

Traditional ASR pipeline:

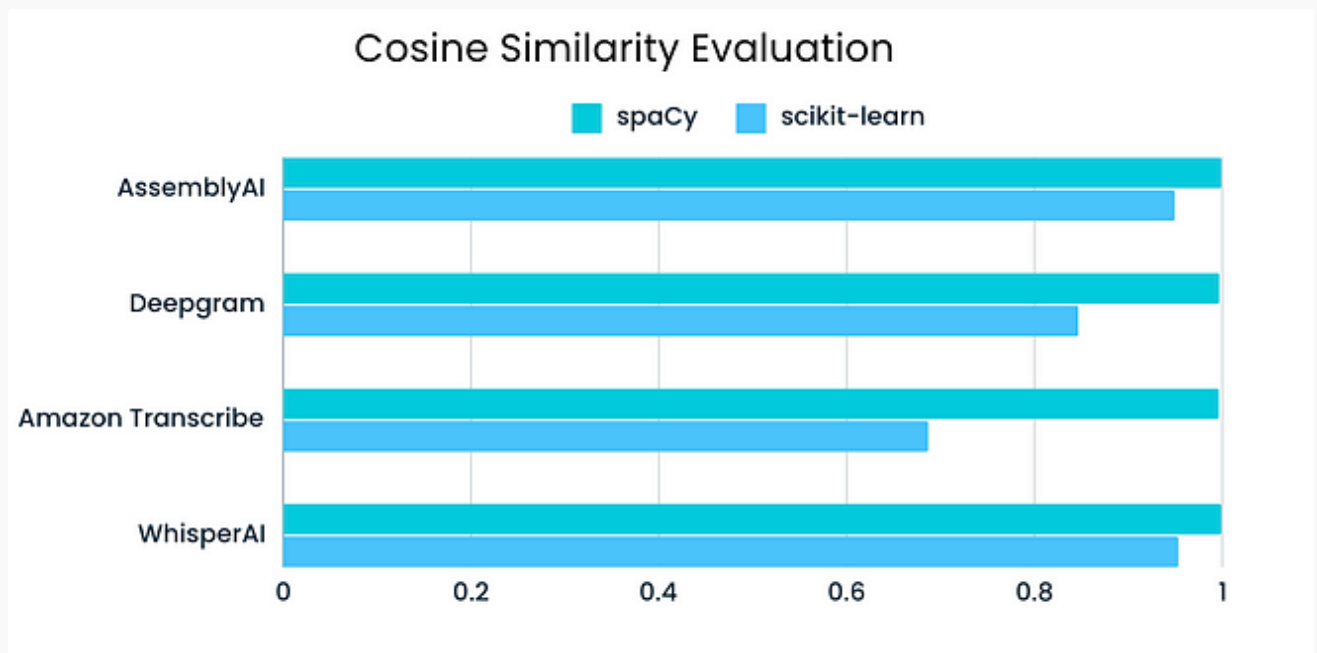


Ty

The Future of Speech Recognition Is Now

- Introduction: The Evolution of Speech Recognition

★ Mar 6



Anna Kiefer

It Started With a Whisper

A Comparison of Popular Speech-to-Text Services

Feb 5  14  3



Amazon.com

Software Development Engineer

Seattle, WA

Mar. 2020 – May 2021

- Developed Amazon checkout and payment services to handle traffic of 10 Million daily global transactions
- Integrated Iframes for credit cards and bank accounts to secure 80% of all consumer traffic and prevent CSRF, cross-site scripting, and cookie-jacking
- Led Your Transactions implementation for JavaScript front-end framework to showcase consumer transactions and reduce call center costs by \$25 Million
- Recovered Saudi Arabia checkout failure impacting 4000+ customers due to incorrect GET form redirection

Projects

NinjaPrep.io (React)

- Platform to offer coding problem practice with built in code editor and written + video solutions in React
- Utilized Nginx to reverse proxy IP address on Digital Ocean hosts
- Developed using Styled-Components for 95% CSS styling to ensure proper CSS scoping
- Implemented Docker with Seccomp to safely run user submitted code with < 2.2s runtime

HeatMap (JavaScript)

- Visualized Google Takeout location data of location history using Google Maps API and Google Maps heatmap code with React
- Included local file system storage to reliably handle 5mb of location history data
- Implemented Express to include routing between pages and jQuery to parse Google Map and implement heatmap overlay



Alexander Nguyen in Level Up Coding

The resume that got a software engineer a \$300,000 job at Google.

1-page. Well-formatted.



Jun 1



11.4K



156



See more recommendations