Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
April 7, 2022
Project Report

Final Project Report

Spoken Language Classification

Group 1: Rubert Martin & Achaebe Parker & Graham Smith & Max Henry & Emmanuel Wilson

ECSE Department, McGill University, Montreal , H3A 0C6, Canada

Abstract

Motivation: In this project the authors experiment with deep learning architectures as applied to the problem of spoken language classification. The main techniques investigated are transfer learning, autoencoding for domain transfer, and prototypical networks. All approaches take as a basis a ResNet50 model which is pretrained on ImageNet.
Results: Baseline transfer learning achieves the highest test accuracy, though analysis of confusion matrices reveals how prototypical networks differentially affect the representation space in a way that may be more flexible for few-shot learning.
Availability and Implementation: Code is available on GitHub at hub.com/dgsmith1988/ECSE-552-Final-Project/. The datasets are available at http://www.repository.voxforge1.org/downloads/SpeechCorpus/.

1 Background and Motivation

In this project, the authors sought to pursue neural audio learning on a practical scale. The team was motivated first and foremost to choose an audio task; this goal was tempered by the fact that it should be possible within a reasonable period of time, with limited compute resources.

Deep learning applications in audio can be broadly considered in two categories: (1) neural signal processing, which includes text-to-speech (i.e. audio synthesis conditioned on words, as in [15■]), denoising [19■], and timbre transfer (i.e. musical synthesis conditioned on pitch and loudness, as in [5■]); and (2) audio understanding, which includes audio classification [18■] and speech comprehension [21■] . The former can be considered as a regression problem, in which each output audio sample is a prediction—this demands, at standard sampling rates for neural learning, at least 16-thousand predictions per second. The latter category is comparably much less demanding: audio classification, for example, amounts to one prediction per input.

Language classification [14■] is an important preliminary step for many secondary audio understanding applications, for example in speech-to-text transcription. The meaning of speech sound is entirely language-dependent: e.g., the sound "/d/-/on/-/k/" means 'so,' in French, and is a slang term for 'to hit,' in English. While the problem of language identification is not new, the task has been recently well-established with a known benchmark on a publicly available dataset [20■] . In this work, the authors achieve an 89% accuracy identifying one from six languages (English, Spanish, French, German, Italian, and Russian) taken from the VoxForge dataset[1]. The current state of the art on this task comes from Shore et al. [21■] who include the task in a benchmark suite that encourages multi-task learning for audio representations. The authors achieved an accuracy of 94.1% on the same six-language dataset.

Digital audio is recorded and consumed as one-dimensional time-series, a format that is not well-suited to fully-connected neural networks. Earlier attempts at audio understanding made use of temporally-aware network structures, such as long short-term memory (LSTM) networks [25■] Audio can also be represented in two-dimensions, via so-called time-frequency transformations. Such transformations represent audio over time (typically presented along the x-axis) and frequency (y-axis), and resemble a 1-channel image. Such a representation makes audio data amenable to image-type processes, such as 2D convolutions. In 2017, Hershey et al. [8■] applied popular convolutional neural network (CNN) architectures to audio classification, selecting ResNet50 [7■] as the most successful model.

The benchmark paper for this task ([20■]) uses a Resnet50 that was pretrained on the ImageNet [3■] dataset. ImageNet has 1000 classes of natural images, including for example: "bookshop," "dog," and "warplane." We pick this model as our baseline, with the goal of exploring the effect of alternative training regimes on resulting model performance and latent space.

2 Problem definition

The language classification task typically comprises identifying one of six languages: English, Spanish, French, German, Italian, Russian. Our goal is to attack this problem within the constraints of a limited time and computer budget. Therefore we limit our task to identification of three languages: English, Spanish and German. Building on top of a pretrained ResNet50 baseline, we explore how the prediction accuracy and computation speed

[1] http://www.voxforge.org

vary for alternative training regimes. Audio samples from the VoxForge Dataset (see Section 2.1) are classified depending on the language utilized by the speaker. The authors pursue three different approaches:

- A pretrained ResNet50 network, with tuned hyperparameters (Section 3.2).
- A ResNet50 fine-tuned with an a autoencoder pretraining regime (Section 3.3).
- A prototypical network based on fewshot learning (Section 3.4).

We report the following metrics for all approaches, to facilitate comparing the relative strengths and weaknesses of each:

- Number of epochs needed to surpass 95% prediction accuracy (during training and validation).
- Prediction accuracy (on a held-out test set).
- Confusion matrix for classification of three languages (on a held-out test set).

The first metric, number of epochs needed to reach 95% accuracy, is meant to give an insight into the practical aspects of model trainability; a lower value here reflects a model that is relatively fast to train. The value of 95% was chosen to represent an acceptable validation accuracy, that can be expected to fall for an unseen test set. Test prediction accuracy is a global measure of the performance of the particular model in the given task, and reflects its ability to generalize to unseen data. We expect the confusion matrix to gives insight into the learned representation space, particularly in the ability of each model to distinguish between languages by pairs. The three chosen languages (English, Spansh and German) have an interesting relationship: while Spanish is a Romance language, English inherits from both Romance and Germanic language groups. Therefore one might expect models to have a greater confusion between the English-Spanish, and English-German pairs; however this trend may not hold across all models.

2.1 VoxForge Dataset

The data used in this project come from the VoxForge[2] open-source dataset. This dataset is generated by individual user contributions. Each user is supplied a text file containing a series of prompts in a language of their choosing. Users record themselves speaking the prompt on their own recording equipment; the dataset therefore includes a wide variety of audio quality, including professional recording equipment and more broadly available mobile and desktop microphones. The whole dataset contains 17 languages, however the following are the most popular languages by number of submissions: English (6319), French (2260), Spanish (2248), German (1419) and Italian (1060). All recordings are saved as .wav files in a variety of sampling rates and bit-depths, ranging from 8–48 kHz and 16–32 bits respectively. Included in each submission is a transcription of the prompts supplied to the users, as well as information on their recording setup.

To reduce the amount of pre-processing required, we restricted our selection to only files sampled at 16 kHz and a bit-depth of 16. This sample rate is common for neural inference, and even synthesis tasks; however speech communications can go as low as 8 kHz, and therefore, arguably, a human being does not need much spectral information to determine one language from another. A samplerate of 16 kHz limits the spectral information to 8 kHz (by the Nyquist-Shannon sampling theorem). Although there is strong precedent to limit speech content to a sampling rate of 8 kHz to allow for intelligibility, we hypothesized there could be spectral information above 4 kHz which would be useful for language identification.

[2] http://www.voxforge.org

In addition to limiting our selection to three languages, further restrictions had to be made on the dataset to meet the memory limitations provided by our online tools (Google drive, Google colab). We limited the total number of submissions from each language to be a random sample of 1600 users, each user providing a variable amount of individual speech samples. This resulted in the following sample distribution:

- English: 18,903
- German: 37,126
- Spanish: 17,056

Half of this data was allocated for testing, and the other half was split 80/20 for training and validation in each of the following training regimes. Note that the the same training and validation dataset was used in each regime, and each regime made only one inference on the test set.

One further note, regarding the dataset split by speaker. The train and validation dataset comprise 800 speakers for each language with each speaker contributing multiple samples. Importantly, different samples from the same speaker are present in both the train and validation sets. However, the test set uses 800 speakers that are entirely different from the 800 speakers in the train/validation dataset. As a result, models must be able to generalize to a completely unseen set of speakers, which is close to deployment-type conditions.

3 Methodology

3.1 Computational Architecture

Care was taken to design a computational architecture which allows for experimental repeatability, as well as efficiency in computational execution. In order to help achieve experimental repeatability, we chose to use resources which everyone in the group has access to. GitHub is used for storage of the source code, to house a standardized training loop as well as functions to compute common performance metrics. Google Drive stores both the measurement logs as well as the training data, due to the lack of persistent storage available on Google Colab. To increase the speed of the computations, all the data were transferred to the Colab instance at first before any processing was done to ensure no network access commands would be executed during training. By ensuring all the team members were accessing the same data and code. any differences between experimental runs could be isolated to either the Colab configuration (i.e. allocated GPU) as well as random variable initialization (i.e. optimizer starting points or data splits).

3.2 Hyperparameter Tuning

The language classification process presented in this work depends on multiple non-differentiable parameters, which may be tuned to give optimal performance from each of the proposed models. These parameters include the settings to calculate the spectrogram, namely: the number of Mel channels for analysis (which determines the height of the spectrogram), the range of frequencies analyzed, the limit for signal level in decibels, the hop length between analysis windows, and the maximum time after which each audio sampled was clipped before being fed to the network. Certain other learning characteristics, such as batch size and type of optimizer, were also considered for optimization.

To limit the search-space, spectrogram parameters were selected from previous literature ([8, 20, 21■]), and a hyperparameter grid-search was performed on batch size, sample length, and choice of optimizer. The obtained values can be seen in Table 1.

Duration	**Batch size**	**Optimizer**	**n_hop**	**n_fft**
5s	128	Adam	512	2048
n_mels	**f_min**	**f_max**	**Level**	
128	20 Hz	8300 Hz	80 dB	

Table 1. Spectrogram and parameters used across all training regimes.

3.3 Autoencoder

The ResNet50 baseline model is pretrained on data from the ImageNet dataset. This dataset comprises only pictures of real world scenes, rather than spectrograms which will be used in this project. In this approach, we consider an autoencoder setup as part of a pretraining regime to perform domain adaptation: i.e., to shift the latent representation from the domain of natural images, to one that is more appropriate to spectrogram images.

An autoencoder has two main components, an encoder and a decoder. Simply put, the encoder compresses the raw signal information (here, spectrograms) into a compressed latent representation, leaving only the most relevant information to describe the input. (A note that not all encoders are compressing, though they are in this case). The decoder takes a latent representation and attempts to reconstruct the initial data received by the encoder [12■]

In this training regime, we use a pretrained ResNet50 as an encoder, and train a decoder with an inverse Resnet structure. Once this model has converged, we remove the decoder and attach a one-layer prediction head (1000 to 3, for three language classes) onto the latent representation layer. The hypothesis for this pretraining regime is that if the pretrained model is optimized for spectrograms in this way, the classifier will train faster than the baseline network, and may be able to perform better given less data.

The autoencoder architecture was built using the weights from the pretrained ResNet50, replaced inside the Pytorch Lightning autoencoder module. Due to the lack of pretrained ResNet50 *autoencoders*, the decoder for our model was not pretrained. The optimizer used for the pretraining regime was ADAM with a learning rate of 0.001. The reconstruction loss of the model can be seen in Figure 1.

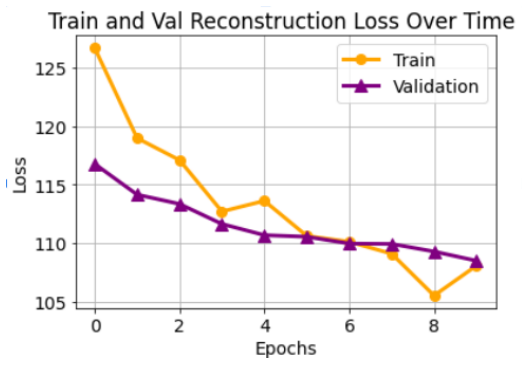

Fig. 1. Training and Validation image reconstruction loss of the autoencoder pretraining regime.

3.4 Prototypical Networks

The prototypical networks algorithm [23■] is a form of metric-distance learning. The algorithm is designed to develop an embedding space that is well-behaved for unseen classes, and as such is a kind of meta-learning; i.e., the space should be informative in the context of few-shot, or zero-shot learning.

As a parameter, the algorithm asks for a support set size, S, which is typically much smaller than the batch size. At each batch, the software collects S samples from each class to make a 'support set' for that class. These samples are projected through the neural network into latent embeddings. The S embeddings from each class are then averaged, making a 'prototype' embedding for that class. For C classes, each batch will generate C prototype embeddings. All other samples in the batch, denoted the 'query set,' are projected through the same network. The algorithm measures the Euclidean distance between each query and the C prototype embeddings. The C distances are negated, then put through a soft-max function. The resulting values constitutes a probability distribution that the query sample belongs to a class; higher values indicate a closer distance to a given prototype vector, which counts as a prediction in favour of that class. The algorithm proceeds in this way, minimizing the cost function as measured by cross-entropy between the proximity-based (i.e., inverse-distance) predictions and the ground truth.

The authors claim that this process is incapable of over-fitting, and indeed the validation accuracy did not seem to drop in our training. We trained with this algorithm using a support size of 2 samples, for 200 epochs without early stopping.

4 Results and Discussion

4.1 Baseline Resnet50

The baseline model was run for 50 epochs with the weights unfrozen. In this way, the pretrained network was allowed to fine tune on the new task, while the prediction head was trained from scratch. The loss and accuracy data were collected at each epoch. (As a note, in here and all following figures, we record the 0'th epoch metrics after one epoch of training; i.e., we have no metrics for the model's very initial prediction, which is expected to be close to random in accuracy (33%). As can be seen in Figure 2, the validation loss drops off quickly in the first 10–15 epochs, before plateauing. In this and all following models, checkpoints are saved automatically for the weights that produce the lowest validation loss, and these weights are used in testing.

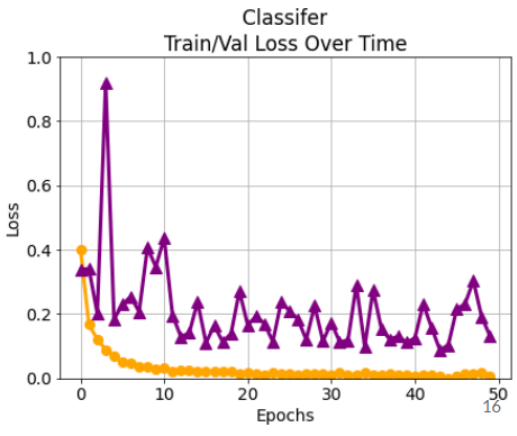

Fig. 2. Baseline model training and validation loss for 3 language classification

As seen in Figure 3, the validation accuracy rose to a height of 95%, in pace with the descending loss function, before plateauing. The checkpoint weights give a validation accuracy of 97.0%. The model reached and surpassed 95% validation accuracy in 12 epochs.

The baseline model achieved a test accuracy of 90.2%. The confusion matrix in Figure 4 reveals some interesting patterns: in particular, the

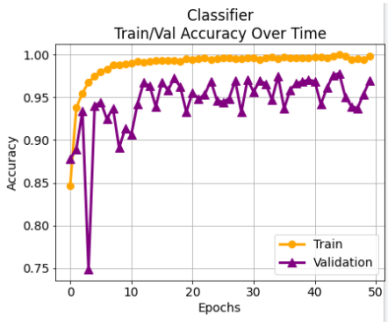

Fig. 3. Baseline model training and validation accuracy for 3 language classification

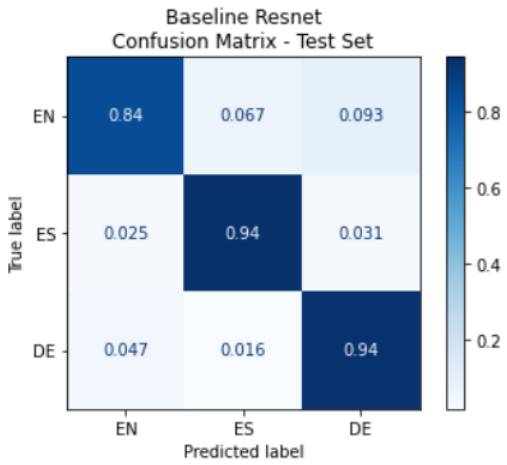

Fig. 4. Baseline model confusion matrix

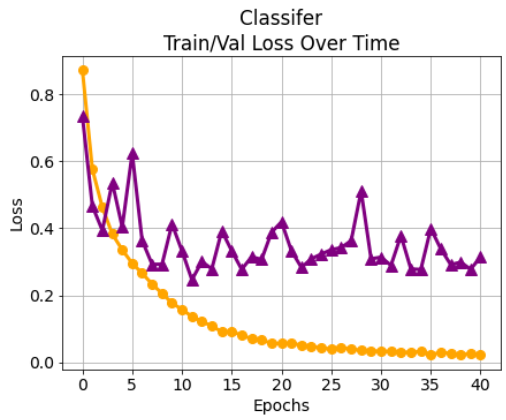

Fig. 5. Training and validation loss of AE-pretrained language classifier.

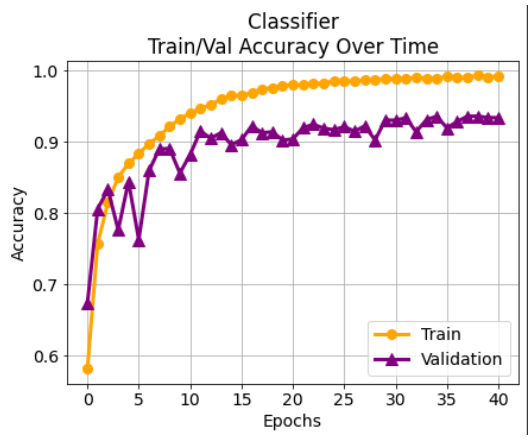

Fig. 6. Training and validation accuracy of AE-pretrained language classifier.

prediction accuracy is highest for the German and Spanish languages, but is 10% lower for the English language. Spanish has Romantic origins, whereas German has (clearly) Germanic origins. By contrast, English has roots in both Romantic and Germanic languages. From this perspective, it is sensible that the network confuses English for Spanish and German, while Spanish and German have a greater distinction from one another.

These results will be the baseline to which the other two models will be compared against.

4.2 Autoencoder

The autoencoder-pretrained model was trained for 40 epochs, with the training/validation loss and accuracy logged at every epoch. In comparison to the baseline model, this network takes longer to converge, as its validation loss takes longer to reach a stable level.

As seen in Figures 5 and 6, not only does the validation loss take longer the reach a plateau, but so too does it take for the model to reach a high accuracy. This model reaches a top validation accuracy of 93.6%, and therefore forgoes the metric of 'number of epochs to reach 95% validation accuracy.' This result should not necessarily be interpreted to mean that the network trains slowly; in fact, the model appears to reach a pleateau in performance fairly quickly (within 10 epochs). However, the capacity of the model is clearly limited, as it is incapable of achieving a higher validation score.

The AE-pretraining appears to have a detrimental effect on the model, as this regime results in a poorer performance than the baseline model. This result is likely due to the fact that the *decoder* component of the autoencoder was not adequately trained. An example output of the trained autoencoder can be seen in Figure 7. The decoder is a significantly large model (25M parameters), and starts from scratch in the pretraining regime. Since the decoder is untrained at the outset, the autoencoder is not able to reconstruct spectrograms. As a result, this pretraining regime may be 'undoing' the pretraining of the encoder (as the unfrozen network scrambles to learn how to reconstruct images), rather than fine-tuning the representation to be sensitive to spectrogram-specific features, as intended. We believe that the autoencoder would need to be trained for a much longer period of time in order to properly leverage the strength of this training regime. Unfortunately such an experiment is outside of the scope of this work.

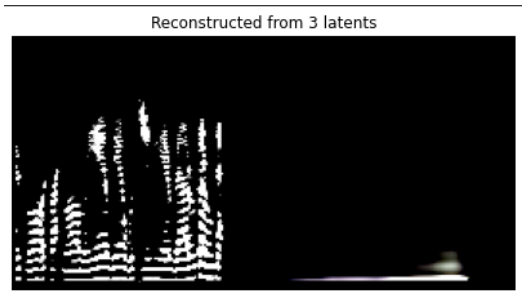

Fig. 7. AE spectrogram reconstruction example. Left, spectrogram input; right, autoencoder reconstruction.

On the unseen test set, the AE-pretrained model achieves a reasonable accuracy of 85.4%. Examining the confusion matrix in Figure 8, one can see that the network is able to make a strong distinction between Spanish and German about as well as the baseline model. The model does outperform the baseline in classifying Spanish, however it struggles with the distinction between English and German. Given the hypothesis that English is an ambiguous language in the context of this task, we suspect that the autoencoder regime, by diminishing the capacity of the pretrained encoder network, has exacerbated this weakness.

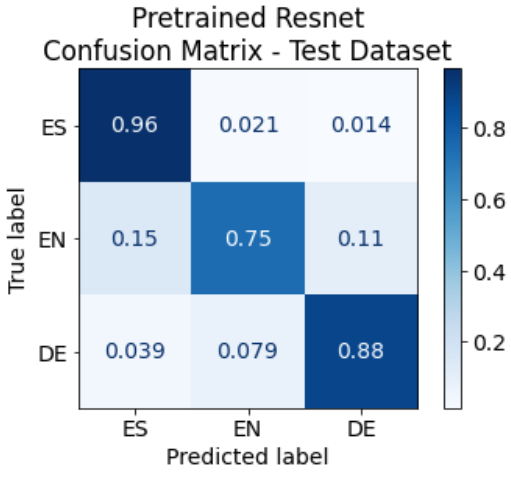

Fig. 8. Confusion matrix of pretrained language classifier.

Technical difficulties. Due to resource and time constraints, the autoencoder was only trained over a period of 10 epochs. As a result of the network comprised of two ResNets (more than 50M parameters), the resources and computational time required to train far exceeded expectations. Provided with the proper hardware, pretraining would need to run for much longer in order to optimally fine-tune the pretrained ResNet.

The most notable issue was due to the pretrained model having been developed using color images while the spectrograms are effectively monochromatic. The pretrained networks expect inputs consisting of three dimensions (corresponding to the RGB layers) so the output of the spectrograms had to be expanded. This alone causes the memory footprint of the data to be increased threefold. In preliminary attempts at dimensional expansion, the data was also converted from as single-precision float to a double-precision which when combined with the extra dimensions multiplied the memory requirements/usage by six. Before refactoring, this limited the batch size to one which given the complexity of the model and Google Colab's limits on computational time, renders the model computationally infeasible.

Interestingly, this highlights the need for understanding the implementation aspects of the models being worked with as well as the training methods used. Depending on where the computational points are defined, what the data dependencies are, and how much of the computations can be parallelized, it's quite easy to see how the run-time memory usage for a simple model increases substantially beyond just its weights for the forward and backward passes. For a much more complex model, it would be even worse as there would likely be a higher number of intermediate calculations to store.

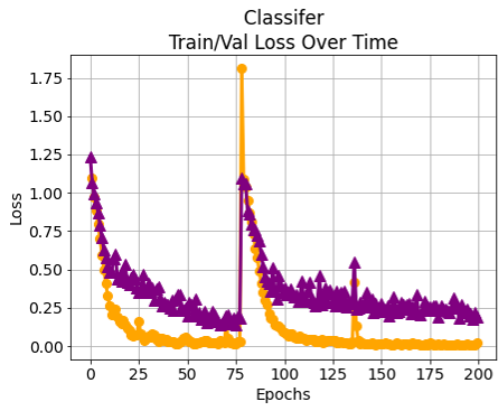

Fig. 9. Prototypical networks train and validation loss.

4.3 Prototypical Networks

The prototypical network was trained for 200 epochs without early stopping. Training and validation loss (Figure 9), as well as accuracy (Figure 10), were recorded for at each epoch. Despite multiple runs, the network exhibited a consistent bug in which the loss and accuracy reset at some time between 70 and 80 epochs. Regardless, by 200 epochs the model appears to regain its asymptotic performance, reaching a final validation accuracy of 96.02%. As predicted, and excepting the bug in the software, the validation loss never dipped, suggesting that the network did not overfit to the training data.

The network achieved a test accuracy of 83.0% on the held-out testing set. Figure 11 shows a confusion matrix reflecting the performance of the model. In comparison to the other models, performance is fairly even across classes, which reflects the well-conditioned nature of the learned space. However, a trade-off for this well-calibrated performance is the fact that the network did not achieve nearly the same levels of test accuracy as the other models. We also note that in the training phase, it takes a considerably longer time for the model to achieve a reasonable validation accuracy. The prototypical network takes 50 epochs to reach 95% accuracy on the validation set, while it takes the pretrained baseline model a 12 epochs to achieve this performance (see Table 2).

The even performance between all classes suggests that this network may indeed be more performant in a few-shot or zero-shot setting, where it has little *a prioi* access to example classes before inference time. The trade-off is two-fold, however: (1) it takes a much longer compute time to get the model to a sufficient level, and (2) once this level is achieved, a well-conditioned metric space may represent a compromise, i.e., it may compromise inference accuracy on classes where a bit of class-conditioned bias (which is trained out of the representation space in this regime) may be useful. For example, in this dataset it may be that the Spanish class, which has the greatest accuracy in the other models, is inherently easier to identify (perhaps due to an unintended bias in the dataset, e.g., recording bandwidth (quality)).

Training regime	**Test accuracy**	**Epochs to 95% val.**
Baseline (fine-tune)	**90.2%**	**12**
Autoencoder	85.4%	N/A (>40)
Prototypical network	83.0%	50

Table 2. Overview of model and algorithm performance.

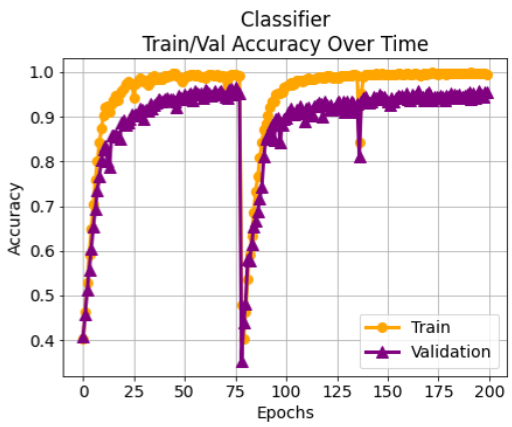

Fig. 10. Prototypical networks train and validation accuracy.

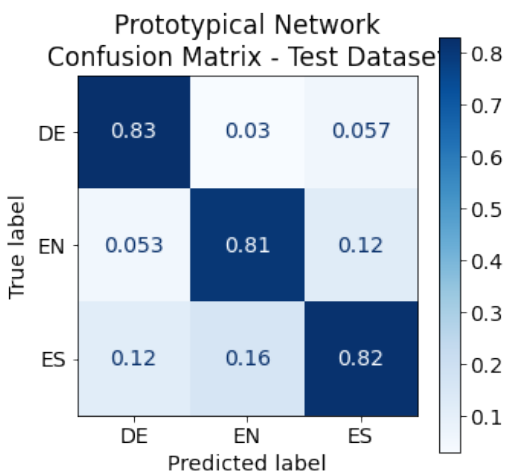

Fig. 11. Confusion matrix for prototypical networks performance on test set.

5 Conclusions and Future Directions

5.1 Conclusions

Language classification is an important task that has the capacity for many practical applications. In this case, three languages (English, Spanish German) were chosen to be classified. Audio files from these languages were turned into spectrograms in order for common deep learning algorithms could be trained on this dataset.

A baseline model, created from a pretrained Resnet50, showed a 90.2% classification accuracy for these three languages. An autoencoder, also utilizing a Resnet50, classified these languages with an accuracy of 85.4%. Finally, the prototypical network achieved a classification accuracy of 83.0%. The baseline and autoencoder models confused English with other languages more often, while the prototypical network achieved a lower but more even performance across each language.

Overall, the autoencoder and prototypical models performed almost as well as the baseline model. The autoencoder approximately matched the baseline when classifying Spanish, but fell short while classifying the other two languages. Meanwhile, the prototypical network approximately matched the baseline when classifying English, but fell short while classifying the other two languages.

Each training regime explored has shown a differential effect on the resulting latent space that is learned; this result is clear from the analysis of the confusion matrices, which show that each model has its own particular strengths. Even though each experiment used the same dataset and the same model architecture, the choice of pretraining and or training regime can have a significant impact on the results of the prediction, and different regimes may be desirable for different applications. For a quick and reliable result, one could do worse than by fine-tuning on a reliable pretrained dataset, especially one that has been trained for many days or weeks, and one whose application in the target domain is well recorded and well-known. We can conclude from this work three principle findings: (1) model architecture isn't everything, and pretraining has a significant effect on the result of the model, (2) relying on literature and precedent can be an important time-saver when applying your own neural networks to short term problems in practical contexts, and (3) sometimes extra effort, fancy training regimes, and additional pretraining can cause damage to a simpler, more effective model. While it is important to work with a model whose capacity is up to the challenge of the task proposed, it is also important not to get in the way of this model.

5.2 Future Directions

There are many different possible aspects of this work which could be extended and investigated further. One reasonable first step would be extending the work here to six languages, making for easier comparison to the existing corpus of literature on this task. This expanding would also provide a good opporutnity to see how the models extend to unseen languages, which might demonstrate the superiority of the metric space learned by prototypical networks. There are other, more pragmatic directions as well. Determining a lower bound on the sampling rate for a particular performance level is one possible extension. A lower sampling rate would decrease the amount of processing necessary to make a prediction and improve real-time latency which would be particularly applicable given the rise of various voice activated personal digital assistants (i.e. Echo/Nest). One might also want to explore domain specific knowledge – for example, signal enhancement processing like echo cancellation and de-noising could be explored to see how much they can improve performance. Finally, as the confusion matrices of the two models showed different strengths for different languages, a coherent framework for exploring ensemble methods that combines the strengths of each model is another potential avenue for future exploration.

Acknowledgements

The group would like to acknowledge the TAs and professor for their help in choosing a direction and other support throughout the project.

References

[1] *Audio Classification using FastAI and On-the-Fly Frequency Transforms*. URL `https://towardsdatascience.com/audio-classification-using-fastai-and-on-the-fly-frequency-transforms-4dbe1b540f89`.

[2] Dor Bank, Noam Koenigstein, and Raja Giryes. "Autoencoders". In: *arXiv:2003.05991* (2020)

[3] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[4] Shubham Dokania and Vasudev Singh. "Graph Representation learning for Audio & Music genre Classification". In: *arXiv preprint arXiv:1910.11117* (2019)

[5] Jesse Engel et al. "DDSP: Differentiable digital signal processing". In: *arXiv preprint arXiv:2001.04643* (2020)

[6]Jing Han et al. "Sounds of COVID-19: exploring realistic performance of audio-based digital testing". In: *NPJ digital medicine* 5.1 (2022) pp. 1–9.

[7]Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[8]Shawn Hershey et al. "CNN architectures for large-scale audio classification". In: *ICASSP* IEEE 2017, pp. 131–135.

[9]Andrew G Howard et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications". In: *arXiv preprint arXiv:1704.04861* (2017)

[10]Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012)

[11]Ignacio Lopez-Moreno et al. "Automatic language identification using deep neural networks". In: *ICASSP* IEEE 2014, pp. 5337–5341.

[12]Andrew Cunningham Ferenc Huszar Lucas Theis Wenzhe Shi. "Lossy Image Compression with Compressive Autoencoders". In: *ICLR* (2017)

[13]Brian McFee et al. "librosa: Audio and music signal analysis in python". In: *Proceedings of the 14th python in science conference*. Vol. 8. Citeseer. 2015, pp. 18–25.

[14]Gregoire Montavon. "Deep learning for spoken language identification". In: *NIPS Workshop on deep learning for speech recognition and related applications*. Citeseer. 2009, pp. 1–4.

[15]Aaron van den Oord et al. "Wavenet: A generative model for raw audio". In: *arXiv preprint arXiv:1609.03499* (2016)

[16]Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[17]Adam Paszke et al. "Pytorch: An imperative style, high-performance deep learning library". In: *Advances in neural information processing systems* 32 (2019)

[18]Jordi Pons, Joan Serrà, and Xavier Serra. "Training neural audio classifiers with few data". In: *ICASSP* IEEE 2019, pp. 16–20.

[19]Dario Rethage, Jordi Pons, and Xavier Serra. "A wavenet for speech denoising". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* IEEE 2018, pp. 5069–5073.

[20]Shauna Revay and Matthew Teschke. "Multiclass language identification using deep learning on spectral images of audio signals". In: *arXiv:1905.04348* (2019)

[21]Joel Shor et al. "Towards learning a universal non-semantic representation of speech". In: *arXiv:2002.12764* (2020)

[22]Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014)

[23]Jake Snell, Kevin Swersky, and Richard Zemel. "Prototypical networks for few-shot learning". In: *Advances in neural information processing systems* 30 (2017)

[24]*VoxForge*. URL `voxforge.org`.

[25]Jianchao Zhou. "Sound event detection in multichannel audio LSTM network". In: *Detection and Classification of Acoustic Scenes and Events (DCASE)* (2017)

[26]Jie Zhou et al. "Graph neural networks: A review of methods and applications". In: *AI Open* 1 (2020) pp. 57–81.