Machine Learning Engineer Nanodegree Capstone Project

Environmental Sound Classification

Akshay Bhardwaj

November 2nd, 2018

I. Definition

Project Overview

The problem of environmental sound classification is a prominent area of research with it having applications in a platitude of areas, ranging from context aware computing and surveillance to noise mitigation enabled by smart acoustic sensor networks. It is even utilised in content-based multimedia indexing and retrieval.

A variety of signal processing and machine learning techniques have been applied to the problem, including matrix factorization [1], dictionary learning [2], wavelet filterbanks [3] and most recently deep neural networks [4]. The most recent techniques have involved using deep Convolutional neural networks (CNN) [5], and have been particularly successful as they are capable of capturing energy modulation patterns across time and frequency when applied to spectrogram-like inputs, which has been shown to be an important trait for distinguishing between different, often noise-like, sounds such as engines and jackhammers [6].

I was intrigued by this problem after learning about the work done by Google Brain researcher Sara Hooker in this field. She worked with non-profit organizations in Kenya, utilizing old mobile phones to identify chainsaw noises in Kenyan rainforest and thus helping reduce illegal deforestation.

I will be using the audio dataset URBANSOUND8K for this project, which can be found here:
https://urbansounddataset.weebly.com/urbansound8k.html

Problem Statement

The goal of this project is to evaluate the performance of deep-learning CNNs for the classification of urban sonic events. We will try to utilize the power of image classification of CNNs to do this by using the novel approach of turning audio files to spectrogram images and then using transfer learning to classify them. This is done because we want to utilize the recent advancements made in image classification and use transfer learning. We will be using the Resnet34 CNN architecture for the classification task. Moreover, we will be using 10-fold cross validation using the predefined folds provided in the dataset to measure the performance of the classifier.

Datasets and Inputs

We will be using the URBANSOUND8K dataset for this project, which can be found here: https://urbansounddataset.weebly.com/urbansound8k.html

This dataset contains 8732 labelled sound excerpts (<=4s) of urban sounds from 10 classes: air_conditioner, car_horn, children_playing, dog_bark, drilling, enginge_idling, gun_shot, jackhammer, siren, and street_music. The classes are drawn from the urban sound taxonomy.

In addition to the sound excerpts, a CSV file 'UrbanSound8k.csv' containing metadata about each excerpt is also provided. This file contains meta-data information about every audio file in the dataset.

The meta-data contains 8 columns.

- slice_file_name: name of the audio file
- fsID: FreesoundID of the recording where the excerpt is taken from
- start: start time of the slice
- end: end time of the slice
- salience: salience rating of the sound. 1 = foreground, 2 = background
- fold: The fold number (1–10) to which this file has been allocated
- classID:
  0 = air_conditioner
  1 = car_horn
  2 = children_playing
  3 = dog_bark
  4 = drilling
  5 = engine_idling
  6 = gun_shot
  7 = jackhammer
  8 = siren
  9 = street_music
- class: The class name: air_conditioner, car_horn, children_playing, dog_bark, drilling, engine_idling, gun_shot, jackhammer, siren, street_music.

All excerpts are taken from field recordings uploaded to www.freesound.org. Every recording was manually checked by listening to it and inspecting the user-provided metadata. Only those recordings were kept that were actual field recordings where the sound class of interest was present somewhere in the recording. Next, the audio clips were sliced into short audio snippets for sound source identification of the desired classes. They were then allocated to 10 different folds. Some of the excerpts are from the same original file but different slice. If one slice from a certain recording was in training data, and a different slice from the same recording was in test data, this might increase the accuracy of a final model falsely. Thanks to the original research for which the dataset was created, this has also been taken care of by allocating slices into folds such that all slices originating from the same Freesound recording go into the same fold. So there is no need to split the data again into training/validation/testing sets.

Evaluation Metrics

Since this is a multi-class classification problem, several evaluation methods can be used for model evaluation. We will be using the recommended evaluation metrics of the dataset: accuracy. (https://urbansounddataset.weebly.com/urbansound8k.html)

The accuracy will be calculated via 10-fold cross validation using the predefined folds.

10-fold cross validation using the predefined folds: train on data from 9 of the 10 predefined folds and test on data from the remaining fold. Repeat this process 10 times (each time using a different set of 9 out of the 10 folds for training and the remaining fold for testing). Finally report the average classification accuracy over all 10 experiments (as an average score + standard deviation, or, as a boxplot).

This can be represented as a formula as:

$$accuracy = mean\ of\ individual\ CV\ accuracy + std.dev.$$

$$accuracy = \frac{1}{n}\sum_{t=1}^{n} x + \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x-\bar{x})^2}$$

Where,

x = accuracy of model validated on a single fold and trained on rest

n = number of folds

$\bar{x}$ = mean accuracy over all folds

II. Analysis

Data Exploration

We will be using the URBANSOUND8K dataset for this project, which can be found here: https://urbansounddataset.weebly.com/urbansound8k.html

This dataset contains 8732 labelled sound excerpts (<=4s) of urban sounds from 10 classes: air_conditioner, car_horn, children_playing, dog_bark, drilling, enginge_idling, gun_shot, jackhammer, siren, and street_music. The classes are drawn from the urban sound taxonomy.

In addition to the sound excerpts, a CSV file 'UrbanSound8k.csv' containing metadata about each excerpt is also provided. This file contains meta-data information about every audio file in the dataset.

The meta-data contains 8 columns.

- slice_file_name: name of the audio file
- fsID: FreesoundID of the recording where the excerpt is taken from
- start: start time of the slice
- end: end time of the slice
- salience: salience rating of the sound. 1 = foreground, 2 = background
- fold: The fold number (1–10) to which this file has been allocated
- classID:
  0 = air_conditioner
  1 = car_horn
  2 = children_playing
  3 = dog_bark
  4 = drilling
  5 = engine_idling
  6 = gun_shot
  7 = jackhammer
  8 = siren
  9 = street_music
- class: The class name: air_conditioner, car_horn, children_playing, dog_bark, drilling, engine_idling, gun_shot, jackhammer, siren, street_music.

All excerpts are taken from field recordings uploaded to www.freesound.org. Every recording was manually checked by listening to it and inspecting the user-provided metadata. Only those recordings were kept that were actual field recordings where the sound class of interest was present somewhere in the recording. Next, the audio clips were sliced into short audio snippets for sound source identification of the desired classes. They were then allocated to 10 different folds. Some of the excerpts are from the same original file but different slice. If one slice from a certain recording was in training data, and a different slice from the same recording was in test data, this might increase the accuracy of a final model falsely. Thanks to the original research for which the dataset was created, this has also been taken care of by allocating slices into folds such that all slices originating from the same Freesound recording go into the same fold. So there is no need to split the data again into training/validation/testing sets.

Exploratory Visualization

Any multi-class classification task needs to have a balanced dataset for a rigorous model creation. Let's take a look at the number of samples each class has throughout the dataset.

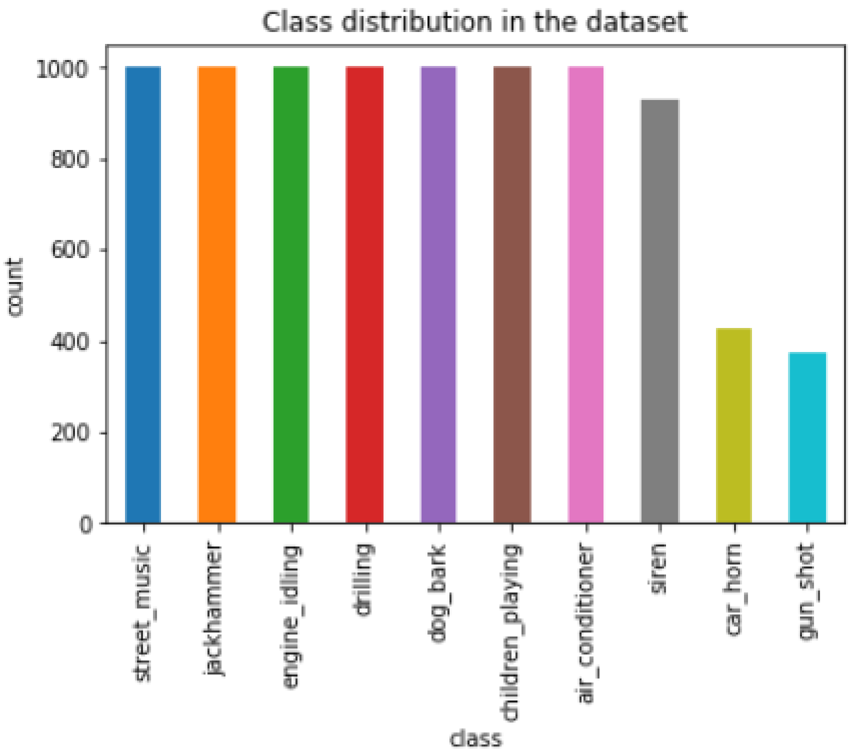


Fig. 1 - Class distribution in the dataset

The bar chart in Fig. 1 shows that the dataset is not perfectly balanced, with the classes siren, car_horn and gun_shot having lesser entries as compared to the other classes. But, it doesn't look like the dataset is severely unbalanced, so no data augmentation for the minority classes will be needed.

Next, let's take a look at the length of the various audio samples.

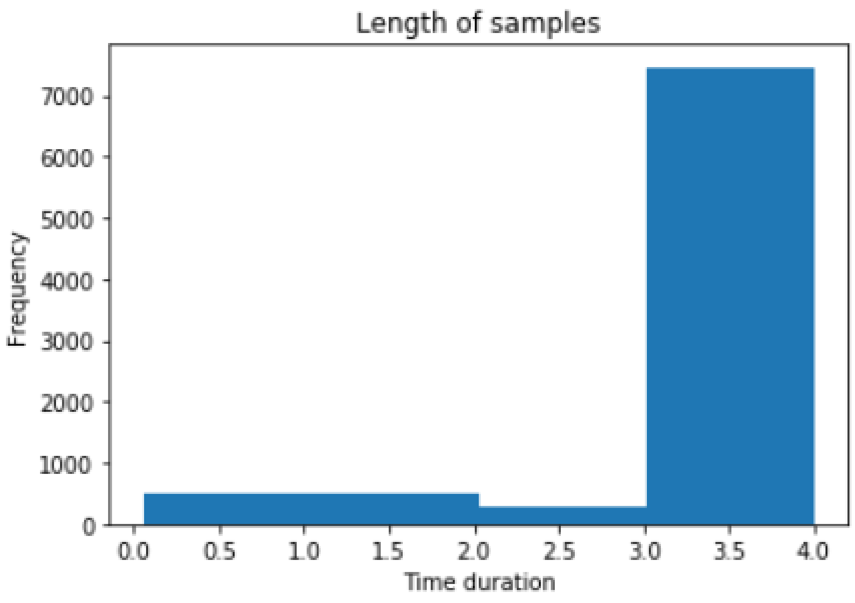


Fig. 2 – Histogram showing length of audio samples

The histogram in Fig. 2 shows that majority of the audio samples are of 3-4 seconds, which fits well with our planned spectrogram method of classification.

Lastly, since we will be using various pre-defined folds for cross-validation, let's examine the audio sample distribution for the folds, ensuring that all of them have similar number of samples.

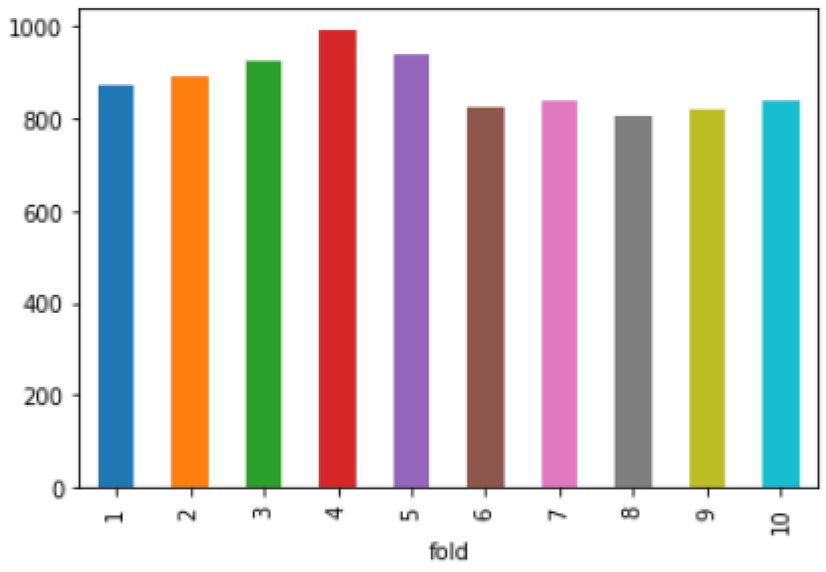


Fig. 3 – Number of samples in each fold

The bar chart in Fig. 3 re-iterates the fact that there are comparable number of samples in each fold.

Algorithms and Techniques

We will be taking advantage of the recent advancement in computer vision to tackle this problem of environmental sound classification. This will be done by initially pre-processing the audio files available in the dataset to their respective Spectrogram images.

Once the Spectrogram images have been created, we will feed them through a deep Convolutional Neural Network (CNN) architecture. The usage of a CNN is motivated by the fact that, unlike the conventional neural nets, they are able to learn relevant features from an image.

Instead of creating the CNN architecture from scratch, we will take advantage of the breakthroughs made in computer vision by using a pre-trained Resnet34 architecture. (It will be retrained later, details covered in the Implementation section below).

The Resnet (residual network) architecture introduces shortcut connections into a feed forward network, allowing us to create deeper networks without degrading the accuracy and error rate. This is done by introducing identity mappings through the shortcut connections. Shortcut connections are those skipping one or more layers, as shown in Fig. 4.

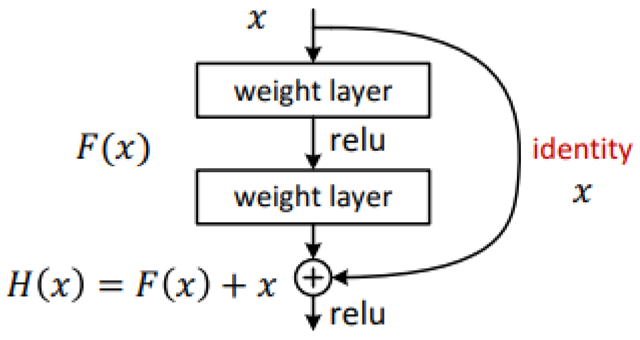


Fig. 4 – Shortcut connections in Resnet architecture

Such small residual networks are stacked together to arrive at a deeper neural network architecture. One such architecture 'Resnet 34' uses the shortcut connections over 34 stacked layers, as shown in Fig. 5, and is utilized for classification by us.

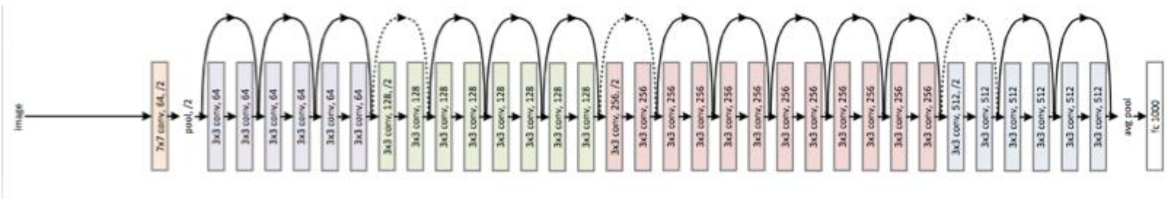


Fig. 5 – Resnet 34 architecture

Benchmark

We will be using the same Resnet 34 architecture for the benchmark model, albeit with a little caveat. The benchmark model will be using a pre-trained Resnet model on the CIFAR-10 dataset and no tuning of its parameters will be done. We will only be tuning the last few layers of the architecture with the Spectrogram images, keeping the weights for all the other layers frozen.

We achieved an accuracy of 65.7% using this untuned model. We will try to do much better than this by tuning the parameters of the architecture in accordance with the spectrograms created.

Moreover, according to the latest publication on the dataset's website, the state-of-the-art mean accuracy achieved for classification was 79%, which was achieved with extensive audio specific data augmentation, and without augmentation their top accuracy was 74%. We will try to beat this accuracy as well.

III. Methodology

Data Preprocessing

Converting the audio files to spectrograms is an important data preprocessing step required here, but we need to keep certain characteristics of the audio files in mind while doing so.

An audio signal is a continuous analogue signal, and it is impossible for computers to process this type of continuous analogue data. It first needs to be transformed into the series of discrete values, and “sampling” is doing just that. “sampling rate” and “bit depth” is two of the most important elements when discretizing audio signal. In the Fig. 6, you can see how they are related to analogue to digital conversion. In the graph, the x-axis is time, the y-axis is amplitude. “sampling rate” decides how frequent it will take samples, and “bit depth” decides how detailed it will take samples.

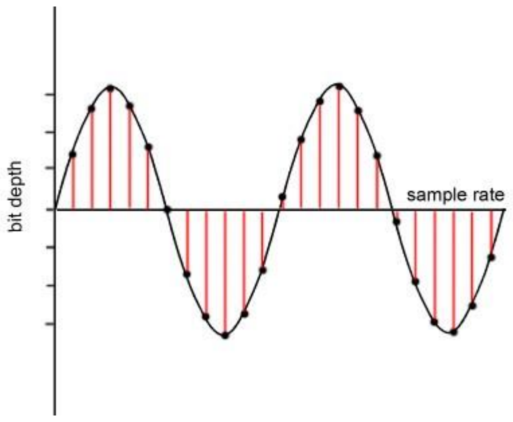


Fig. 6 – Audio Sampling

Additionally, we have to keep in mind the number of channels in each audio files (1 for mono, 2 for stereo).

And according to the description of the dataset, the sampling rate, bit depth and the number of channels are the same as those of the original files uploaded to Freesound, and hence may vary from file to file. This is not good, as it will not lead to a uniform Spectrogram creation. We need to ensure that a uniform value is maintained for all three characteristics for the Spectrograms to be of uniform standard. This can be done in various ways, most notably using techniques such as resampling via the scipy library, but we will take advantage of the functionalities available in the Librosa library to do so.

By default, the Librosa’s load function converts the sampling rate to 22.05khz, as well as reduces the number of channels to 1(mono), and normalise the data so that the values will range from -1 to 1. And thus all our worries about different files having different characteristics are taken care in one go.

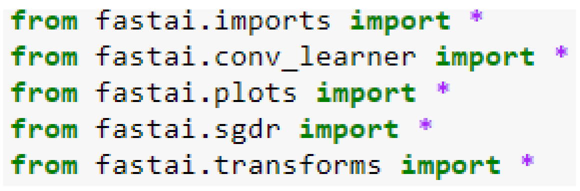
The librosa library also allows us to create mel-scaled spectrograms of the loaded audio files by calling its librosa.feature.melspectrogram method, thus converting our audio files to their respective Spectrograms. We can now use any image classifier to do its classification.

Note: We will not perform any data augmentation on the converted spectrograms because any changes made in the spectrogram, such as zooming, tilting, etc may correspond to other sound categories and hence may make our classifier weaker by giving it false images of categories.

Implementation

After pre-processing the audio files to their respective spectrograms, let's arrange the converted images in the ImageNet style directory structure for each fold. This is done because it allows for easy data loading in the fastai library we will be using for classification.

The fastai library allows for training fast and accurate neural nets using modern best practices. We will start by importing a few of the functionalities of the library:



```
from fastai.imports import *
from fastai.conv_learner import *
from fastai.plots import *
from fastai.sgdr import *
from fastai.transforms import *
```

We then worked on the data created above, using the data from the folds 2-10 for training and the data from fold 1 for validation. We will use the training schedule and architecture we obtain by working on this set of data for the modelling for other folds as well. We will keep training until we see that the CNN starts overfitting.

We start by building a CNN model using the pre-trained Resnet 34 architecture and then training only the last few layers, keeping the initial layers frozen with the pre-trained weights, for 4 epochs. This allows our last layers to get accustomed to the new training data (as it was previously trained with the CIFAR-10 data). This is followed by unfreezing of all the layers of the model and then training the whole model with differential learning rates. The differential learning rates are used because we want to take advantage of the pre-trained weights of the model.

We know that the initial layers of the CNN capture simple features such as horizontal/vertical/slanted lines, colour gradients, etc. and thus they do not need to be retrained with the new images as vigorously as the later layer which capture complex shapes and patterns by combining interactions between the previous layer outputs. This is why we will start by using differential learning rates with lower learning rates for the initial layers of the model and comparatively higher learning rates for the later layers.

After training for a few epochs with the differential learning rate, the model gets a little accustomed to the new images (spectrograms) and then we will switch over to a higher learning rate for all the layers so as to speed up the training process. We then trained the model for a number of epochs, until we see that the model starts to overfit to the data. Overfitting is detected by observing the training and validation losses of the model after each epoch and is characterised by seeing an increase in the validation loss while the training loss is reduced. At this point, we consider the model to be trained and stop further training process.

This is how we arrived at the following training schedule for our model:

```
def process_fold(fold):
    data = get_data(sz, data_directory/fold)
    learn = ConvLearner.pretrained(arch, data)
    learn.fit(1e-4, 3, cycle_len = 1)
    learn.unfreeze()
    learn.fit([1e-6,1e-5,1e-4], 3, cycle_len = 1, cycle_mult = 2)
    m = learn.fit(1e-4, 14, cycle_len = 1)
    return m, learn
```

We have made it into a function in itself to expedite the training and validation over the 10 folds of data we have. It follows the mechanism of loading in the data, defining a pre-trained architecture as a model, training the last layers with the data for a few epochs, unfreezing all layers, training with differential learning rate for a few epochs and then finally training with a higher uniform learning rate for a number of epochs. The number of training epochs chosen for all the steps are based on the initial experimentation done above.

We automate the training process of all the folds using a for loop, storing the returning CV accuracy of each model. These individual accuracies are then combined using the metric:

$$accuracy = \frac{1}{n}\sum_{t=1}^{n} x + \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x-\bar{x})^2}$$

To obtain the final accuracy of the model.

Refinement

We started with an untuned Resnet 34 model and arrived a highly tuned model after several steps of refinement. All the refinement of the model was done based on the experimentation on the single fold. To do this, the single fold model was repeatedly saved and re-loaded between training steps. The refinement process included using different learning rates as well as experimenting with cycle_len and cycle_mult parameters.

The cycle_len and cycle_mult parameters are used for doing a variation on stochastic gradient descent called “stochastic gradient descent with restarts” (SGDR). Briefly, the idea is to start doing our usual minibatch gradient descent with a given learning rate, while gradually decreasing it using cosine annealing, until we jump it back up to the prescribed learning rate. Refer to the fastai library docs for more details about it: https://docs.fast.ai/

Once we had obtained a refined model for the single fold, we used the same model architecture for training and validating the other folds.

IV. Methodology

Model Evaluation and Validation

As stated above, we will use accuracy as our evaluation metric. The classification model will be evaluated via *10-fold cross validation using predefined splits*. This will enable us to compare our performance to those performed by the author of the dataset.

Our final model was a pre-trained Resnet 34 CNN architecture, which was then trained on data from 9 out of the 10 predefined folds and then tested on the remaining fold. This process was repeated 10 times, each time using a different set of 9 out of 10 folds for training and the remaining fold being used for testing the accuracy. Following this method, we got the following individual accuracies for our model:

Testing fold	**Accuracy (%)**
1	75.57
2	68.44
3	72.99
4	73.83
5	80.27
6	69.99
7	72.50
8	70.73
9	76.20
10	81.13

These individual accuracies are then combined using the evaluation metric of mean accuracy + standard deviation to get a final accuracy of **78.36%** for our tuned model.

Justification

The benchmark model was trained in a similar fashion to the tuned final model (10 times over different folds) and had a final accuracy of **65.78%**.

Whereas, our tuned final model achieved an accuracy of **78.36%**, far outperforming our benchmark model. Based on the improvement of 12.58% in our multi-class classification accuracy, the final tuned model can be deemed as a satisfactory solution.

Moreover, according to the latest publication on the dataset's website, the state-of-the-art mean accuracy achieved for classification was 79%, which was achieved with extensive audio specific data augmentation, and without augmentation their top accuracy was 74%.

Since, we have not used any data augmentation method, our model easily performs better than the accuracy of 74% achieved by the state-of-the-art model. More impressive is the fact that even without using any data augmentation, our models performance is only 0.64% worse than that of the state-of-the-art models performance with augmentation. This further reaffirms that our model is performing well.

V. Conclusion

Free-form Visualization

A comparison of the benchmark and final models over various different folds is as shown below:

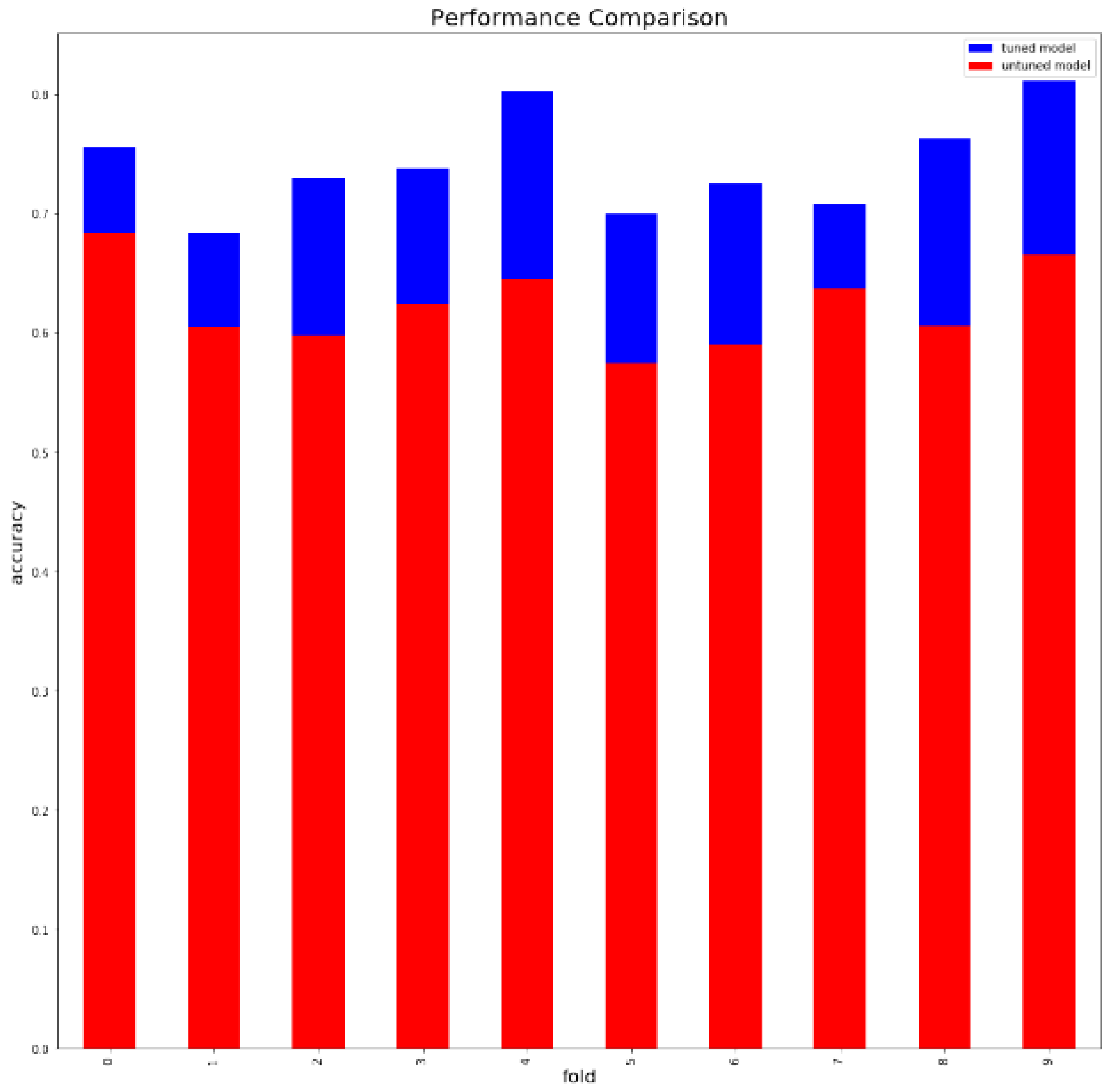


Fig. 7 – Comparison of benchmark and final model over folds

From the plot in Fig. 7 it is clearly visible that our model not only has better overall performance, but also a better performance across each individual fold as well.

Reflection

The process used for this project can be summarized using the following steps:

1. Deciding to do audio classification by converting short audio clips into images.
2. Downloading the dataset and then untaring it.
3. Initial data exploration and visualization to affirm that the dataset is balanced.
4. Preprocessing the audio files by converting them into their respective spectrogram images.
5. Reorganizing the data from different folds in the style of ImageNet directory for easy data loading in classifiers.
6. Visualizing some of the converted spectrograms.
7. Creating a benchmark model with pre-trained Resnet 34 architecture.
8. Tuning the Resnet model and training it to get a final model which performs much better than the benchmark model.
9. Visualize the final performance.

I found the step 4 to be pretty difficult and was not sure on how to use scipy library to normalise the audio clips in accordance with their sampling rate, channel, etc. Thankfully, librosa had this itself, saving me a lot of time.

The tuning aspect in step 8 was very interesting as I saw that even with the same model and training schedule being used for all folds, some of the models tended to converge much more quickly than a few other ones.

Since the model reached very close (only 0.64% less accurate) to the state-of-the-art approach on the dataset, I am quite happy with its performance.

Improvement

A few of the ways the solution can be further improved:

1. Use audio data augmentation techniques before creating spectrograms to generate more training data.
2. Use the newer Inception model for transfer learning.
3. Train the model with a lower learning rate and for a few more epochs.

References

[1] - V. Bisot, R. Serizel, S. Essid, and G. Richard, “Acoustic scene classification with matrix factorization for unsupervised feature learning,” in IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Shanghai, China, Mar. 2016, pp. 6445–6449

[2] - J. Salamon and J. P. Bello, “Unsupervised feature learning for urban sound classification,” in IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), Brisbane, Australia, Apr. 2015, pp. 171–175.

[3] - J. T. Geiger and K. Helwani, “Improving event detection for audio surveillance using gabor filterbank features,” in 23rd European Signal Processing Conference (EUSIPCO), Nice, France, Aug. 2015, pp. 714– 718

[4] - K. J. Piczak, “Environmental sound classification with convolutional neural networks,” in 25th International Workshop on Machine Learning for Signal Processing (MLSP), Boston, MA, USA, Sep. 2015, pp. 1–6.

[5] - J. Salamon and J. P. Bello, “Deep Convolutional Neural Networks and Data Augmentation for Environmental Sound Classification” in IEEE Signal Processing Letters, November 2016

[6] - ——, “Feature learning with deep scattering for urban sound analysis,” in 2015 European Signal Processing Conference, Nice, France, Aug. 2015