AUDIO DEEPFAKE RECOGNITIONDOCUMENTATION

M Aneesh | aneeshmukkamala@gmail.com | +91 7416829748

I have chosen the following 3 approaches for comparison with the ResNet based method for implementing.

1) ResNet-Based Detection Model (Hybrid Feature-based detection)

Due to the deep CNN architecture the models have better feature representations that are sensitive to artifacts often introduced by deepfake synthesis. Uses a learnable filter bank to extract features from spectrograms of the waveforms of the audio files.

LCML (Log-Cosh Loss function) maximized inter-class separability compared to traditional binary cross entropy. This method reported low error rates across which is notable for such a lightweight approach.

Equal Error Rate (EER): 1.81% in the Logical Access (LA) scenario (ranked 4th), t-DCF (tandem detection cost function): 0.052 in LA (ranked 4th)

Of all the approaches this method the least latency making it highly usable. Fine tuning them would be much simpler, it would simply involve stacking other layers or blocks on top of the existing model and the performance would not be compromised.

2) LCNN-Based Detection Model (Handcrafted Feature-based detection)

While the ResNet was forced to learn the artifacts present in fake audios, this method is the opposite, it extracts features that make an audio more genuine/real. Features are extracted based on the Constant-Q Transform (CQT) to obtain a log power spectrum (LPS), though log spectrum is not used in ResNet based approach, I included it as well in my ResNet implementation.

Standard cross-entropy (CE) loss was used. Error-rare is higher than ResNet approach:
EER: 4.07% in LA (ranked 11th) , t-DCF: 0.102 in LA (ranked 10th)

Compared to ResNet this performs relatively better under background noises and real-world situations but the due to higher EER that of the ResNet-based model, this is one of the reasons why I chose not to implement this approach.

3) Transformer-Based Approach (End-to-end detection models)

Uses self-supervised models (Wav2Vec2). Unlike the above approaches, there is no need for audio processing while training the models. These models perform significantly better even when the data is just raw audio. The methods are complex with better and reproducible results.

Due to this EER is the 1.08%, lowest of all approaches.

This uses a dedicated comparable loss function like ResNet and uses Differentiable Architecture search for training the model. The raw audio is converted to representations suitable for transformers using sinc-in time low pass frequency filters, there is no need of resampling, truncating/padding the examples of the dataset which is done in the above the 2 approaches. Due to this the dataset can be very diverse without much concerns of outliers. Being a transformer with attention blocks, it is a constraint to train such compute intensive models. Optimizing for parallel inference would become a challenge.

Common point: Data from ASVspoof 2019 dataset is used in all these approaches so comparing them is reasonable

My analysis and insights from adopting the ResNet approach.

1) Challenges encountered and remedies:

1) Due to compute and memory constraints, I was able to only download smaller datasets which are about 12Gb in size compared to the extremely large datasets that are present in the shared repo which were used for training the larger models.
2) Despite the limited amount, the quality of the dataset I have chosen is of good standards without any jitters and focuses more on the voice and required no cleaning/masking of the background noise.
3)Another reason for choosing this dataset is to avoid class imbalance as current image models are reliable if such case arises while classification
4)In this dataset, the files are balanced in terms of gender of speaker, volume of audio and sample rate of waveform of the audio file.

2) Assumptions made:

Upon analysing the data, I found out that for a single example there are 2 files with the same name but with different audio labels (fake vs real). This led to my main assumption that these audios may or be surely linked to a single video source. I did not use LCML (Log-Cosh Metric Learning) loss function, I used binary cross entropy (BCE) to make it computationally efficient and as my model architecture is fairly minimal. LCML is designed for reducing variance between classes, but instead of depending on a loss function for achieving this, I processed and chose the dataset in a way to avoid using this loss function, also BCE took lesser time than LCML in terms of computation.

3) Reason for selecting this approach:

To keep things fairly lightweight and given the scale of the dataset, I wanted to customize the model architecture to be able to converge faster and not overfit. Another reason is the inference speed. I used 2 different variations of the ResNET.
1) One where I used ResNet18's pre trained residual blocks followed by some fine tuning by freezing model's layers.
2) Second where I implemented a model with same functionality from scratch but with just 2 residual blocks

As reported on multiple sources, ResNet provided the perfect balance between the training time, inference speed and accuracy. Choosing this gave me little to no errors while I was implementing.
Hence, the training was done in 10-15 minutes on a dataset of 5,000 examples and no GPU was used for the model that I made from scratch with accuracy of 96% on validation sets. The inference speed has minimal latency and can be run and deployed easily on CPUs.

4) **How the model works:**
I made the model from scratch and the forward pass of a single example is fairly simple. Taking example and illustrating the changes in shape it undergoes, this is how it is done. The forward pass is identical to the actual ResNet / other RawNet architectures, just the number of residual blocks vary.

1) 1d tensor of shape (1 , seconds * sample rate) is converted to 2d shape of (1 , n_mels , (seconds*sample rate)/hop_len)) where n_mels , hop length are explained in the code.
2) This 2d tensor is padded through Conv1 followed by max pooling to result a 3d tensor
3) This 3d tensor is passed through the Residual blocks, while the 3d shape is maintained, the
4) Average pooling reduces spatial dimensions making it again into 1d of shape (512,1,1)
5) This is passed through fully connected layer (512, num_classes) to produce tensor of shape (num_classes,1)
6) After applying SoftMax on the logits, class with highest probability is chosen.

Performance results:
With less than 20 epochs, the training and validation accuracy reached to 95.94 and 92.57% respectively

Observed strengths:
As mentioned earlier, this approach has smoother loss convergence and requires less training with easier adaptability to any time of data. For multi-lingual purposes, this would be much easier to implement.

5) **Weaknesses and future improvements:**
The main drawback is that when the examples chosen have a lot of background noise or no noise at all (mute audio in between), the performance is not accurate. To handle this, I intentionally set some time steps in the audio waveform to 0 (mute) and the performance was significantly improved. With larger datasets, the models can handle a broader variety and generalize better. To handle class imbalance a weighted loss function can be used in which lesser weight is given to overall loss from classes which are larger in number to avoid underfitting on minority classes.

Questions to address

1. What were the most significant challenges in implementing this model?

Choosing the right architecture to run without GPUs needed careful reviewing of the existing approaches and picking the best one for the given dataset. The original datasets were of high yet domain specific quality and adopting this model to real world use cases where there is a lot of channel distortions and noise is challenging. For example, on two different datasets having such noise, its's more than difficult to make a decision on which data would be best for the model as the model's behaviour is unpredictable

2. How might this approach perform in real-world conditions vs. research datasets?

This approach would not give an out of the box working model and would may not perform up to the benchmarks. This approach is focused more on real-world conditions where classification is simpler and hence was designed for such purposes, at the core for reduced inference speed and much attention was not given on the dataset specifications. Incorporating in research conditions which have more complex audio files, the models should be fine-tuned or even made again from the ground up with better datasets

3. What additional data or resources would improve performance?

Enhancing the training process, the data, including augmentations, multi-lingual examples in the same dataset are always helpful but I have one approach in mind.
Apart from these, applying dedicated time frequency masking, but based on the model's activations can help. For example, a spatio-temporal method can be adopted just to study the timesteps of the audio waveform around which the logits of the model classify the example as real or fake). Such timesteps could then be masked and making the model to even more robust and make the model's feature extraction even more strict. So, attention would be given to fine grained relations between the time steps as for even for a single 3 second audio file with a 16kHz sample rate, there are 48,000-time steps and in larger audio files, the models would not capture the relations well. Implementing such spatio-temporal analysis would improve the models to classify longer and complex audio files. Lastly, access to larger GPUs would surely allow experimenting such deeper methods to actually understand how the models behave while processing audio files.

4. How would you approach deploying this model in a production environment?

From what I have observed, lightweight ResNet models implemented from scratch, dedicated to a specific use case can be deployed easily without any accelerated compute. To break it down, heavy architectures like LCNNs are very well engineered and fine tuning them on audios of various languages would work very well. Instead of this, domain/language specific models with ResNets powering them can be developed for this type of audio analysis which run at a fraction of time of what such heavy models require. for example, in NLP, the Helsinki models (which are a collection of lightweight models) solved the task of language translations with least possible inference time, same way language specific ResNets for deepfake or other audio purposes can be easily synthesized due to the low training time.

The main advantage with such low inference time is that, during inference time multiple forward pass (batched) for a single example would highly benefit by applying majority polling or even using techniques like monte-carlo dropout to maximise the chances of getting the correct answer out of the model.

Quantized models with lowered precision can be used for training & inference to optimize inference speeds.

Further, the model I have trained is deployed on a non-GPU server on huggingface spaces to display the speed and flexibility of the approach I have chosen. I made a minimal and simple streamlit UI for this

I have attached a couple of audio files which can be tested on the application I have deployed.

The app can be accessed here: https://huggingface.co/spaces/aneeshm44/Momenta

Apart from the documentation, I have explained the code in detail in a line by line spelled out manner in comments side by side, especially the functions that process and prepare the data for training. I have also explained the hyperparameters and args used for such functions, how altering them would change the performance and the reasons behind.

x----------x----------x