REAL-TIME NEURAL VOICE CAMOUFLAGE A REPRODUCIBILITY STUDY

Isaac Dunford (id1g19@soton.ac.uk), Cristin-Bianca Botea (cbb1u19@soton.ac.uk)

Yash Yadav (yy8g19@soton.ac.uk)

ABSTRACT

This report has been written for the COMP6258 Reproducibility Challenge where we have taken an accepted paper from the ICLR 2022 conference and attempted to assess the reproducibility of the paper. The original goals for this project were to reimplement the model according to the paper specification and compare it to the code made publicly available on the GitHub. However, due to several absences in the original paper and difficulties in the code provided, this wasn't possible.

1 PROBLEM ANALYSIS

1.1 PAPER REVIEW

The paper we chose to study titled "Real-Time Neural Voice Camouflage" by Chiquier et al. (2021), develops an approach for predictive attacks for "camouflaging" a person's voice and reducing the effectiveness of ASR (Automated Speech Recognition) systems such as DeepSpeech. The paper is the first to be able to achieve real-time performance by choosing the attack which will be the most effective in the future.

In short, the authors of this paper achieves this by training a neural network to create an attack which will maximize the expected loss of an ASR system, such as DeepSpeech, when trying to detect the words that were recorded from the speaker. The neural network comprises of many convolutional layers which can be used in a single feed-forward computation to forecast an attack based on 2 seconds of input speech.

The authors then test their performance using the WER (Word Error Rate), to assess the corruption of the overall sentence, and CER (Character Error Rate), to assess the corruption of the individual words. They benchmark their approach against other adversarial attacks such as uniform noise and gradient descent and test their model against various enhancements of DeepSpeech as well as Wav2Vec2 to demonstrate the success of their model.

1.2 REPRODUCIBILITY APPROACH

In order to assess the reproducibility of the paper, we attempted to verify that the paper is valid and that we can support the conclusions that the authors reach. 3 different approaches were taken to do this:

1. Reimplementation of part of their model according to their specification
2. Download and implementation of the code from the project GitHub[1]
3. Code review to ensure the implementation of the theory from the paper has been performed

A full reimplementation of their code isn't feasible due to the complexity of their model and the time constraints of this project. As their model maximizes the expected loss of another neural network, this would mean training a new version of DeepSpeech on the appropriate LibriSpeech dataset whilst also writing our own way to import and preprocess all of the data when ultimately it is more important to verify the new logic provided in the paper.

[1] https://github.com/cvlab-columbia/voicecamo?search=1

In this paper, the model is trained on 8 NVIDIA RTX 2080 Ti GPUs for approximately 2 days. All the code written for this paper can be found on the Computer Vision Lab at Columbia University GitHub.

Due to resource constraints, this reproduction plans to train the models on only one GPU. In order to accurately assess the reproducibility of the paper we aimed to train both the model created by following the paper and the authors' model on the same machine, and compare their results.

2 IMPLEMENTATION

2.1 DETAILS

2.1.1 REPRODUCING THE MODEL

Creating the model described in the paper was relatively straightforward. The model is made of 8 down-sampling convolutional blocks, 4 up-sampling convolutional blocks and a linear layer. The structure of each block is clearly explained, and all the necessary values are provided, with the only exception being the size of the kernel they used. For the reproduction, a kernel with dimensions of 5x5 was used, fortuitously aligning with the dimensions of the kernel utilized in the original paper. The optimisation function, the loss and all their parameters were also provided.

This model is then integrated into the DeepSpeech model of the original code of this paper, and the latter model gets trained on the specified input. If correctly implemented the results obtained from this implementation should be similar to the ones from the original model. This holds true under the assumption that both models undergo training on an identical computational platform, with identical resource constraints.

The dataset used is explicitly specified, with precise specifications regarding the required size and format of the input. The libraries the authors used are well documented, which facilitates the reproducibility of the model.

2.1.2 TRAINING THE MODEL USING THE AUTHORS CODE

As previously mentioned, the model used in this paper is available on GitHub. The process of training this particular model posed greater challenges than anticipated. This was due to its configuration and the many dependencies it requires.

In order to train the exact same model as in the paper, changes to the files were necessary. These changes consisted of commenting out the experimental segments of code marked with "TO BE EXAMINED" that were not discussed in this paper. Those were mainly 2d convolution layers that use kernels of size 3x3. The right optimization function had to be commented in, as in the repository they use an AdamW optimiser instead of the SGD optimizer mentioned in the paper.

Figure 1 and Figure 2 show the believed structure of the network which had to be inferred from the paper and then the authors' code as numerous parameters are left out of the original paper and the code provided by the author doesn't represent the model they used to get their final results.

Other additions and changes include adding more down-sampling blocks to the right model configuration in order to match the paper's specifications and changing the number of channels of the last down-sampling block from 512 to 256. The name of the configuration file used also needs to be changed as the one that the GitHub repository uses it is not provided.

Furthermore, the requirements given on the GitHub required downloads from the authors' local machines which couldn't be accessed. In addition to this, several versions of the packages in the requirements had to be changed as well as they were conflicting with each other or were outdated. Because of this, retraining the model was not successful.

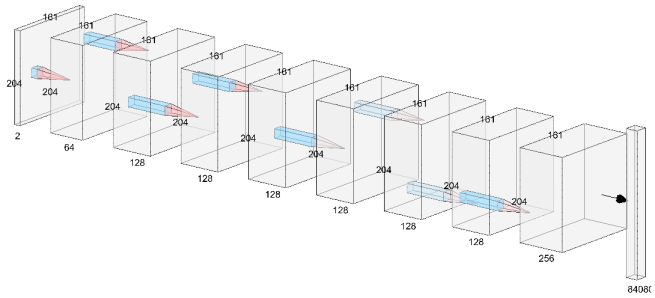

Figure 1: The down-sampling convolutional layers of the network

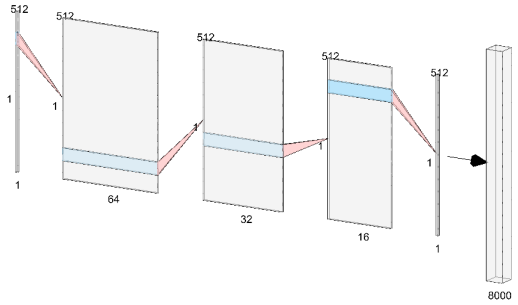

Figure 2: The up-sampling convolutional layers of the network

2.2 Costs

The main costs for assessing the reproducibility of this paper are broken down below:

- Computation - To fully train a model of this size would require a substantial amount of processing power for a long period of time reducing the ability reproduce the paper exactly.
- Development Effort - The paper leaves lots of details about the implementation which can only be found within the code on the GitHub. Furthermore, lots of experimentation was clearly done with the code in the GitHub and the precise code they ran to obtain the results in the paper aren't available.

3 Experimental Results

Due to configuration and requirement-related challenges, neither the original model nor the modified version underwent training. Consequently, assessing the reproducibility of the paper based on model performance is infeasible. Nevertheless, we contend that adopting this approach would have been suitable for evaluating the reproducibility of the paper.

4 Code Analysis and Verification

While analyzing the code several differences were observed, in addition to the aforementioned differences mentioned in section 2.1.2. Notably, certain aspects were found to be unmentioned in the paper. Specifically, the paper describes the construction of all four up-sampling blocks as comprising a unidimensional ConvTranspose layer and a leaky ReLU activation function. However, the implementation uses a stride of 2 in these blocks, an architectural detail not explicitly specified. Furthermore, the final block in their implementation employs a tanh activation function instead of

the specified leaky ReLU function. In addition to this the kernel size in the different layers varies from 5 to 3 without a clear indication as to why.

The provided Python files, clearly show that further work has been done on top of what is described in the accompanying paper, but with the right changes, it can be brought to the desired state for reproduction.

5 CONCLUSIONS

From a strictly analytical standpoint, the model appears to be readily reproducible by adhering to the architectural specifications expounded in the paper. Nonetheless, due to the absence of training outcomes, conclusive assertions regarding the reproducibility of the paper cannot be made. This is attributable to potential concealed optimization steps that may arise from diverse implementation approaches, which are often informed by experiential knowledge in the field, and might not be readily discernible to a novice observer.

However, retraining the existing model of the authors turned out to be more of a challenge due to the conflicting requirements and lack of clear structure in the

REFERENCES

Mia Chiquier, Chengzhi Mao, and Carl Vondrick. Real-time neural voice camouflage. 12 2021. URL https://arxiv.org/abs/2112.07076v2.