GAN-BASED GENERATION AND AUTOMATIC SELECTION OF EXPLANATIONS FOR NEURAL NETWORKS

Saumitra Mishra[*], **Daniel Stoller**[*], **Emmanouil Benetos**[*] [#], **Bob L. Sturm**[†] **& Simon Dixon**[*]

[*]School of EECS, Queen Mary University of London, UK
[#]The Alan Turing Institute, UK
[†]Speech, Music and Hearing, KTH Royal Institute of Technology, Sweden
{saumitra.mishra, d.stoller, emmanouil.benetos}@qmul.ac.uk,
bobs@kth.se, s.e.dixon@qmul.ac.uk

ABSTRACT

One way to interpret trained deep neural networks (DNNs) is by inspecting characteristics that neurons in the model respond to, such as by iteratively optimising the model input (e.g., an image) to maximally activate specific neurons. However, this requires a careful selection of hyper-parameters to generate interpretable examples for each neuron of interest, and current methods rely on a manual, qualitative evaluation of each setting, which is prohibitively slow. We introduce a new metric that uses Fréchet Inception Distance (FID) to encourage similarity between model activations for real and generated data. This provides an efficient way to evaluate a set of generated examples for each setting of hyper-parameters. We also propose a novel GAN-based method for generating explanations that enables an efficient search through the input space and imposes a strong prior favouring realistic outputs. We apply our approach to a classification model trained to predict whether a music audio recording contains singing voice. Our results suggest that this proposed metric successfully selects hyper-parameters leading to interpretable examples, avoiding the need for manual evaluation. Moreover, we see that examples synthesised to maximise or minimise the predicted probability of singing voice presence exhibit vocal or non-vocal characteristics, respectively, suggesting that our approach is able to generate suitable explanations for understanding concepts learned by a neural network.

1 INTRODUCTION

There is an increasing interest in interpreting black-box machine learning models, especially Deep Neural Networks (DNNs) (Doshi-Velez & Kim, 2017). Insights about how models function can assist in gaining trust in their predictions – an essential factor for model adoption in safety-critical applications (e.g. health care, self-driving cars) (Ribeiro et al., 2016b). We can understand a machine learning model by employing one of two strategies. The first involves training inherently interpretable models and is a promising research direction, but often such models perform poorly when compared to state-of-the-art black-box models (Ribeiro et al., 2016a). In contrast, the second strategy involves a post-hoc analysis of a trained model, which does not require compromises on its predictive capacity.

There are two key approaches to bring post-hoc interpretability to DNNs in particular (Montavon et al., 2018). The first focuses on explaining the predictions of a model (Simonyan et al., 2014; Bach et al., 2015; Ribeiro et al., 2016b) for a given input, while the second analyses components (e.g. neurons or layers) of a DNN (Olah et al., 2017). We focus on the second approach in this paper, as it yields general insights about how a DNN forms its predictions.

We can analyse components of a DNN using different methodologies. For example, one can use feature inversion to map latent codes to the input space highlighting the discriminative information a DNN preserves at its layers (Mahendran & Vedaldi, 2015; Dosovitskiy & Brox, 2016). In another direction, one can analyse features that different components of a DNN are sensitive to. One way

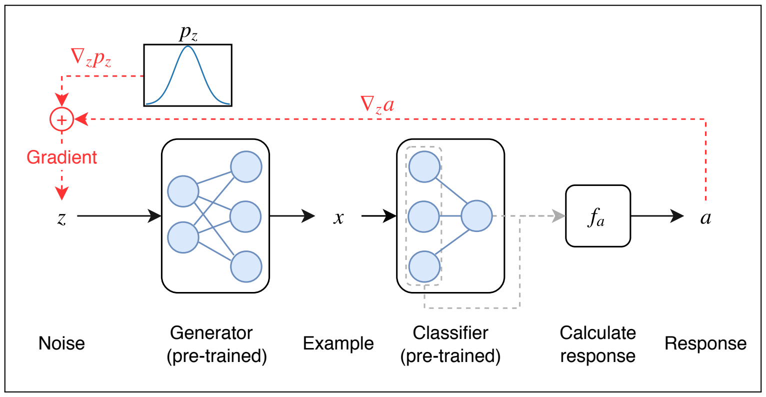

Figure 1: Overview of our proposed approach. A noise vector z is used to generate an example x, for which a response $a \in \mathbb{R}$ is calculated with a response function f_a from all neuron activations of the classifier. f_a can be defined depending on which aspect of the classifier is of interest; examples include the activation of a certain neuron, or the average layer activation. z is optimised to maximise the response a, but also the prior probability $p_z(z)$ to favour realistic outputs.

to do this is by identifying instances from the dataset that maximally activate different components in a DNN (Zhou et al., 2015). Another way is by using Activation Maximisation (AM) (Erhan et al., 2009) that iteratively optimises random noise to synthesise examples in the input space (e.g., images) to maximally activate a neuron or layer in a DNN. Since AM is data-independent and tends to focus more on the explanatory input factors, we will pursue an AM-based approach in this paper.

The interpretability of examples generated by AM depends on two key factors: optimisation of hyper-parameters and the prior. Generally, interpretable examples are selected for each neuron by performing a grid search in the hyper-parameter space and visually inspecting each generated example (Nguyen et al., 2016a), but this is subjective, prohibitively slow and limits the hyper-parameter search space. Also, such an approach is not scalable to analysing other DNN neurons that may require different hyper-parameter settings.

The use of priors for AM restricts the input search space to prevent generating uninformative, adversarial examples. Researchers have proposed several hand-crafted priors for effective AM, synthesising interpretable images (Yosinski et al., 2015; Nguyen et al., 2016b; Mahendran & Vedaldi, 2015). In another direction, Nguyen et al. (2016a) demonstrated that replacing hand-crafted priors by a learned prior (adversarially trained feature inverter) considerably improves the interpretability of synthesised images. However, their approach requires training a separate prior for each layer in the classifier, and appears to rely on the prior and the classifier model having similar architectures.

In this work, we aim to tackle these challenges, making the following contributions:

- To our knowledge, we are the first to use a Generative Adversarial Network (GAN) for example generation using AM. This imposes a strong prior and enables effective AM for any given part of a classifier and even other classifiers with the same input domain without re-training the generator. The work by Nguyen et al. (2017) is closest to ours, in which the authors use a denoising autoencoder as a prior on the latent code of the adversarially trained feature inverter.
- We propose a quantitative measure estimating the interpretability of a set of generated examples by adopting the Fréchet Inception Distance (FID) metric (Heusel et al., 2017). We provide evidence for its effectiveness by qualitatively analysing the synthesised examples.
- We apply our method to a state-of-the-art deep audio classification model that predicts singing voice activity in music excerpts. This results in visualisations that successfully capture the concept represented by the ground truth labels the classifier was trained to predict. There have been some recent works in understanding deep audio classification models, but they either use a different method (Mishra et al., 2018) or perform AM with hand-crafted priors (Zhang & Duan, 2018; Krug & Stober, 2018).

2 Method

Figure 1 provides an overview of our method. For a pre-trained neural network classifier f_c with M neurons and input $x \in \mathbb{R}^d$, our goal is to provide examples that activate a given neuron activation pattern ("classifier response"). Formally, we define $f_n(x) \in \mathbb{R}^M$ as the output activations of *all* M neurons in the classifier f_c for a given input example x. The classifier response we aim to explain can then be defined in a general fashion as the output of some function $f_a : \mathbb{R}^M \rightarrow \mathbb{R}$ that takes *all* M neuron activations of the classifier as input. f_a can be set to output the activation of a single neuron, or the average activation of one or multiple layers, but any differentiable function is supported.

2.1 Activation maximisation

We can perform activation maximisation to find an input example $\hat{x} \in \mathbb{R}^d$ so that the resulting activation $f_a(\cdot)$ is maximised:

$$\hat{x} = \arg\max_x f_a(f_n(x)) \tag{1}$$

The above objective can be optimised by stochastic gradient descent (SGD) by backpropagating through the classifier layers.

2.2 GAN-based prior

However, activation maximisation often produces adversarial examples (Nguyen et al., 2015), which can be very different from inputs encountered during classifier training and testing, are hard to interpret and do not explain the classifier's behaviour for real-world inputs. Furthermore, optimising over the input x directly is often difficult, especially if the dimensionality d is high (Nguyen et al., 2016a).

Our method makes use of a GAN (for more details, see Goodfellow et al. (2014)), where a generator $f_g : \mathbb{R}^n \rightarrow \mathbb{R}^d$ is trained to map a noise vector $z \in \mathbb{R}^n$ drawn from a known noise distribution p_z to a generated example x, and optimises

$$\hat{z} = \arg\max_z f_a(f_n(f_g(z))) + \lambda \log p_z(z). \tag{2}$$

The weighting term $\lambda \geq 0$ is a hyper-parameter controlling the trade-off between activation maximisation and the realism of the generated examples. Note that we search in the low-dimensional noise space for a vector $\hat{z}$ whose associated generator output $f_g(\hat{z})$ produces a high activation, which avoids optimisation issues. To encourage realistic outputs, the real data density p_x should ideally be used in the form of a prior term $\log p_x(f_g(z))$ in equation 2, but we do not have access to p_x. However, assuming a well-trained generator, we can use $\log p_z(z)$ instead, since it should be approximately proportional.

To optimise equation 2 with gradient descent, we require p_z to be a continuously differentiable distribution. Note that this does not include the uniform distribution commonly used for training GANs, for example in (Goodfellow et al., 2014; Radford et al., 2015; Hjelm et al., 2017).

2.3 Example generation

The previous section 2.2 demonstrated how one example is generated in our approach. To generate a set of N examples, we draw N random noise vectors $\tilde{z}_1, \ldots, \tilde{z}_N$ independently from p_z as initialisation points for SGD. The resulting examples should be diverse, so converging to the same optima of equation 2 independent of initialisation is undesirable. Therefore we set the SGD learning rate l_r as well as the number of update steps N_t as hyper-parameters, since they control the influence of the initialisation points on the generated examples and thereby the amount of diversity and randomness.

2.4 Hyper-parameter optimisation

To optimise the prior weight λ and optimisation parameters l_r as well as N_t, it would be ideal to have human subjects evaluate the usefulness of the explanations resulting from different configurations, but this is prohibitively time-intensive. Therefore, we introduce a novel, automatic metric for quickly evaluating a set of generated explanations, allowing efficient hyper-parameter optimisation.

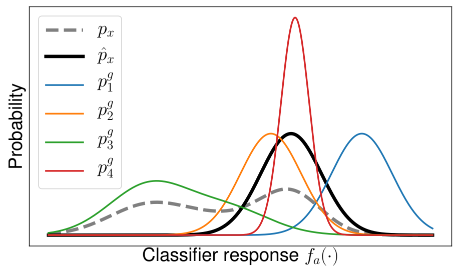

Figure 2: Intuitive explanation for our proposed metric, showing the distributions of activations $f_a(\cdot)$ obtained for input examples from the dataset (p_x), of the dataset examples with the highest N responses $f_a(\cdot)$ ($\hat{p}_x$), and of four hypothetical generators, $p_1^g, \ldots . p_4^g$. Our metric determines which generator distribution is most similar to $\hat{p}_x$ to ensure realistic examples.

In the following, we will explain our reasoning using the hypothetical example in Figure 2. We posit that good interpretability requires the generated examples to have a similar distribution of classifier responses $f_a(\cdot)$ as the N samples with the highest response from the dataset ($\hat{p}_x$ in Figure 2). This is because unrealistic adversarial examples (generator 1 in Figure 2) often lead to large responses. Also, too much weight on the GAN prior λ or ineffective optimisation (generator 3) leads to examples that are realistic, but have too low responses compared to real examples. Additionally, the variance of responses should be similar (making generator 2 the best according to our metric) to ensure a sufficient degree of diversity in the generated samples (in contrast to generator 4).

To take the average and the variance of the responses into account, we therefore adopt the Fréchet Inception Distance (FID) (Heusel et al., 2017) as our distance metric. Since our responses f_a for each example are scalar values, the FID reduces to

$$D((\mu_r, \sigma_r), (\mu_g, \sigma_g)) = (\mu_r - \mu_g)^2 + \sigma_r + \sigma_g - 2(\sigma_r \sigma_g)^{\frac{1}{2}}, \tag{3}$$

where μ_r and μ_g are the means and σ_r and σ_g the unbiased sample variance of the (one-dimensional) real and the generated response distribution, respectively.

In the following section 3, we will investigate whether the metric proposed above adequately reflects how useful the set of generated examples is to a human observer.

3 Experiments

To evaluate our method and to investigate whether approaches based on AM can transfer to domains other than computer vision, we apply it to audio classification. Specifically, we will consider Singing Voice Detection (SVD), a binary classification task (Lee et al., 2018) where a classifier predicts whether singing voice (vocals) is present in a segment of a music recording.

3.1 Choice of classifier

We select a state-of-the-art SVD model[1] introduced by Schlüter & Grill (2015). The model is an eight-layer Convolutional Neural Network (CNN) the architecture of which is mentioned in Appendix (Table 4). It takes a Mel spectrogram of an audio excerpt of 1.6s duration as input. The Mel spectrogram is calculated by first applying an FFT [2] and taking the magnitudes of the resulting spectrum. Later, we apply a Mel filterbank [3] to summarise the energies across different frequency bands. Finally, we normalise the resulting non-negative values by applying $x \rightarrow \log(\max(x, 10^{-7}))$. Using a single neuron with sigmoid activation in the last layer, the CNN predicts the probability of singing voice being present at the centre of the input audio excerpt. Schlüter & Grill (2015) train the model

[1]Available as open source at `https://github.com/f0k/ismir2015`

[2]Using a window size of 1024 and a hop size of 315 samples, with audio sampled at 22050Hz

[3]The Mel filterbank is a set of band-pass filters distributed along the Mel-frequency scale, which is a perceptual scale of pitch defining a logarithmic relationship between frequency and perceived pitch. We use 80 filters in our work, ranging from 27.5 to 8000 Hz.

Table 1: The architecture of our generator. The transposed convolutional layers (ConvT) as well as the FC layer have LeakyReLU activations.

Layer	Input shape	Filter size	Stride	No. of filters/neurons	Output shape
FC	128	-	-	5120	5120
ConvT	$8 \times 5 \times 128$	5×10	2×2	64	$16 \times 10 \times 64$
ConvT	$16 \times 10 \times 64$	5×20	2×2	32	$32 \times 20 \times 32$
ConvT	$32 \times 20 \times 32$	5×20	2×2	16	$64 \times 40 \times 16$
ConvT	$64 \times 40 \times 18$	5×20	2×2	8	$128 \times 80 \times 8$
Conv	$128 \times 80 \times 8$	5×5	1×1	1	$128 \times 80 \times 1$

in a supervised fashion by minimising the binary cross-entropy loss between model predictions and the ground-truth labels.

As training input, the authors randomly sample excerpts from the 93 French Pop music songs contained in the Jamendo dataset (Ramona et al., 2008). The dataset is pre-partitioned into subsets of 61 (training), 16 (validation) and 16 (testing) songs, respectively, and each song has manual annotations indicating the start and end times of each vocal segment.

We replicated the proposed approach of the authors[4], obtaining a classifier whose performance is very similar to the one reported by the authors, as shown in Appendix (Table 3).

3.2 Choice of response function

In this study, we focus on generating positive and negative examples that maximally or minimally excite the final output neuron of the classifier, respectively. Compared to using other definitions of f_a, this allows us to directly evaluate the characteristics of our generated examples, as the positive examples should differ from the negative ones by the presence of singing voice since the classifier is known to be accurate at singing voice detection.

Our initial experiments show that the predicted probability converges to 0 or 1 after only very few iterations, leading to vanishing gradients due to saturation of the sigmoid non-linearity, effectively halting optimisation. We argue this indicates an inherent problem of the classifier and not our method, as neural networks are well-known to be prone to making over-confident predictions (Gal & Ghahramani, 2016). Thus, we applied our method to the pre-sigmoid activations of the final neuron instead.

3.3 GAN training

We use the FMA dataset (Defferrard et al., 2017) for training the GAN, selecting only Pop music pieces to reduce the data complexity and to make the song selection more similar to the one used for training the classifier. The audio signals are converted to Mel spectrograms, replicating the classifier preprocessing described in section 3.1. From each song's full spectrogram, we create clips with 115 time frames and an equal amount of spacing between each clip.

For the generator, we choose a standard normal likelihood $\mathcal{N}(z|\mathbf{0}_n; \mathbf{I}_n)$ for the continuously differentiable noise term $p_z(z)$, with a dimensionality of $n = 128$. The generator architecture is a CNN adapted from the DCGAN (Radford et al., 2015) and is shown in Table 1, using multiple strided transposed convolutions. The final convolution outputs a $128 \times 80 \times 1$ tensor, which is cropped evenly at the borders to obtain 115 time frames as required by the classifier. The final convolution employs $x \rightarrow \max(x, \log(10^{-7}))$ as activation function to ensure the generated spectrogram magnitudes are in the same interval range as the Mel spectrograms obtained from preprocessing real audio samples following section 3.1.

[4]The open-sourced version of the classifier introduced by Schlüter & Grill (2015) is based on Theano and Lasagne and was ported to Tensorflow as part of this work

Table 2: The architecture of our discriminator. Convolutional layers have LeakyReLU activations and bias. The fully connected layer has no bias or activation function.

Layer	Input shape	Filter size	Stride	No. of filters/neurons	Output shape
Conv	$128 \times 80 \times 1$	5×80	2×2	32	$64 \times 40 \times 32$
Conv	$64 \times 40 \times 32$	5×40	2×2	64	$32 \times 20 \times 64$
Conv	$32 \times 20 \times 64$	5×20	2×2	128	$16 \times 10 \times 128$
Conv	$16 \times 10 \times 128$	5×10	2×2	256	$8 \times 5 \times 256$
FC	10240	-	-	1	1

The discriminator is again similar to the DCGAN (Radford et al., 2015) and is shown in Table 2, making use of multiple strided 2D convolutions to process the Mel spectrogram input of size $115 \times 80 \times 1$. The output is a scalar real value used to distinguish real from generated samples.

We use the WGAN-GP objective for training our GAN as in (Gulrajani et al., 2017), with a GP weight of 10. The Adam optimiser with a learning rate of 10^{-4} is used to train the generator and discriminator for 600,000 iterations with a batch size of 16.

3.4 AM Optimisation

We perform a grid search over our hyper-parameters, using $l_r \in \{0.1, 0.01, 0.001\}$, $\lambda \in \{0.1, 0.01, 0.001\}$ and $N_t \in \{100, 500, 1000\}$, giving 27 possible settings. We sample $N = 50$ noise vectors from the noise distribution p_z, resulting in $N = 50$ examples along with their respective activation values $f_a(\cdot)$ for each setting after applying our method. Also we feed the training dataset to the classifier and record the last neuron activation for each excerpt, and select the top $N = 50$ excerpts with maximum activation. We generate a new excerpt of 115 consecutive Mel spectrogram frames ($\widehat{=}$1.6 sec) for every 50 time frames ($\widehat{=}$0.7 sec) in a recording. We optimise our objective in equation 2 by using the Adam optimiser with $\beta_1 = 0.99$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$.

4 Results

In section 4.1, we analyse the effectiveness of our metric proposed in section 2.4 in the context of hyper-parameter optimisation, before employing the best configuration to evaluate our explanation generation system in section 4.2.

4.1 Hyper-parameter optimisation

We investigate whether our evaluation metric reflects the quality of the results obtained for different hyper-parameter settings. Figure 3 shows the GAN output for two randomly sampled initial noise vectors $\tilde{z}_1, \tilde{z}_2$ and the corresponding results after maximising the last neuron activation using three different hyper-parameter configurations C_1, C_2 and C_3. In our best configuration C_1, only small changes are made to the initial output, which ensures diverse and realistic outputs due to the random values of $\tilde{z}$ and high likelihood under the prior p_z, while still increasing the response effectively. The median configuration C_2 leads to further increased harmonic as well as high-frequency energy, although to a slightly unrealistic extent, and configuration C_3 produces extremely sparse outputs with maximum energies one magnitude greater than those found in real spectrograms, and unrealistically high responses of up to 78, thus is ineffective. This shows our metric can rank the different hyper-parameter settings with respect to how useful the resulting explanations are for a human observer. Further perceptual studies are left for future work to establish a stronger connection.

4.2 Qualitative analysis of explanations

We use the configuration C_1 (Table 5 in Appendix) to produce positive and negative examples for the last output neuron of our vocal classifier by maximising or minimising its activation, respectively. Since the classifier was trained to distinguish vocal from non-vocal audio, this allows us to investi-

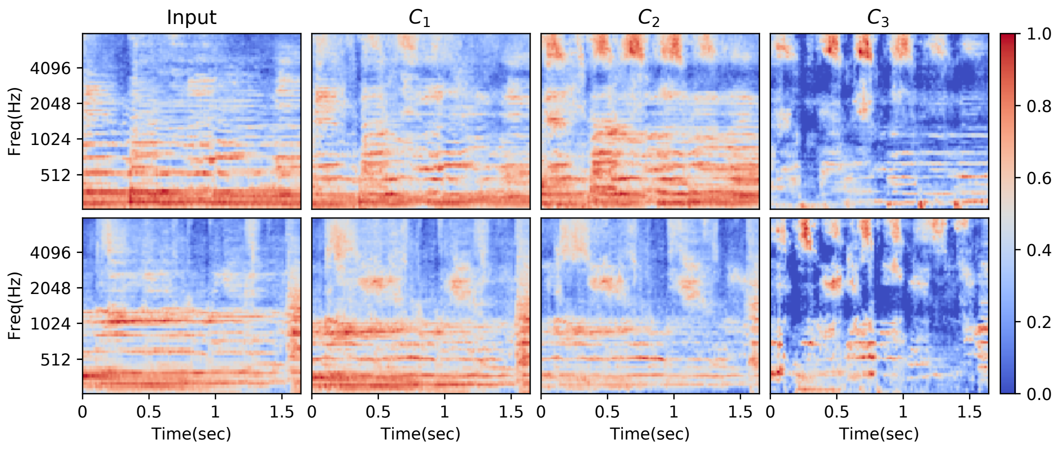

Figure 3: Mel spectrogram visualisations demonstrating the effectiveness of our example evaluation metric from section 2.4, each normalised in scale independently so that red colors show relatively high and blue colors show relatively low spectral energy. The leftmost column shows the output of our GAN $f_g(\tilde{z}_i)$ for two initial noise vectors $\tilde{z}_1, \tilde{z}_2$ (one per row). The others show the result of applying our method with hyper-parameter configurations C_1, C_2 and C_3, which represent the best, median and worst configuration from the set of 27 configurations according to our evaluation metric, respectively. For more details about the configurations, refer to Table 5 in the Appendix.

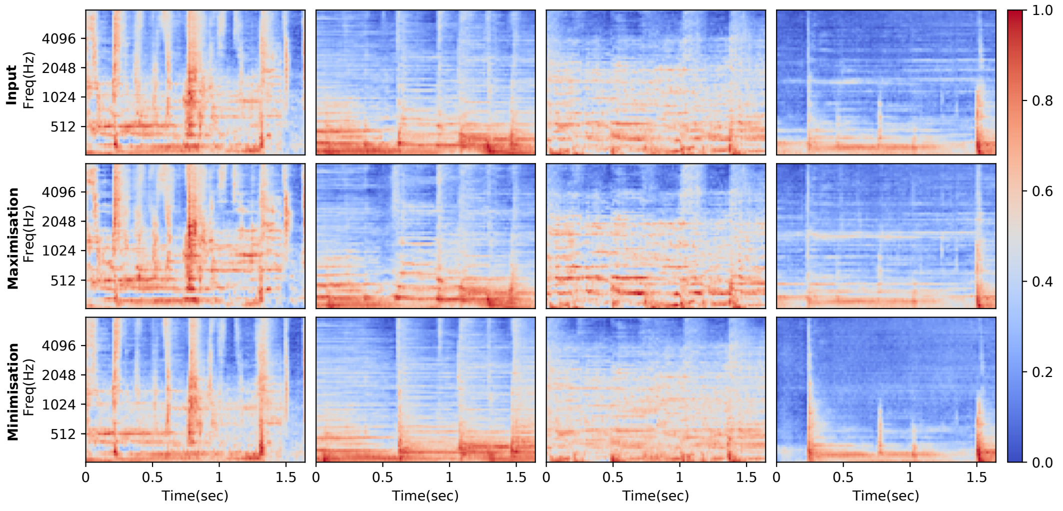

Figure 4: Mel spectrogram visualisations illustrating the concepts the neuron in the output layer of the classifier learns, each normalised in scale independently so that red colors show relatively high and blue colors show relatively low spectral energy. The top row represents initial GAN outputs $f_g(\tilde{z}_i)$ for four initial noise vectors $\tilde{z}_1, \tilde{z}_2, \tilde{z}_3, \tilde{z}_4$. The second and third rows represent examples synthesised by maximally and minimally activating the output neuron. Table 6 in the Appendix details the activations from the last layer neuron for each visualisation depicted above.

gate whether our method can successfully capture the concept of vocal presence using its positive examples, and the concept of pure accompaniment using its negative examples.

Figure 4 shows four pairs of positive and negative examples, each generated using the same noise vector $\tilde{z}$ as initialisation point. We observe a stronger presence of harmonic content in the positive examples, a lack of energy in the very low frequency band below the human voice range, and few transient sounds such as drum hits (visible as vertical bars), indicating our positive explanations indeed have many characteristics typical to vocal content. In contrast, the negative examples have stronger transients and more bass frequency content, indicating the successful generation of purely instrumental examples. Since initial listening tests of the resynthesised audio confirms these observations, this suggests that our GAN-based approach can provide explanations useful for understanding the concepts acquired by a neural network. Furthermore, Table 6 demonstrates that our method effectively optimises the response in all cases. A more quantitative, large-scale listening test is left for future work.

5 Conclusions

In this paper, we presented a GAN-based approach for efficiently generating inputs to a classifier so that its response is maximised, while maintaining realism thanks to its strong prior. Compared to previous approaches, it can be applied more flexibly to new classifiers and to excite different neurons in a classifier. We validated our method on a pre-trained singing voice classifier, showing it can retrieve the concept of singing voice presence encoded in the last output neuron. We presented a metric for automatic evaluation of the usefulness of a set of generated explanations, and use it for optimising the hyper-parameters of our approach. We qualitatively showed that our metric favours the subjectively more interpretable settings. For future work, we plan to conduct listening tests that present generated examples and require the prediction of the model's behaviour for unseen inputs to quantify the interpretability of the explanations.

Acknowledgments

DS is funded by EPSRC grant EP/L01632X/1. EB is supported by RAEng Research Fellowship RF/128 and a Turing Fellowship. This work is supported by EPSRC grant EP/R01891X/1.

References

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PloS ONE*, 10(7), 2015.

Michaël Defferrard, Kirell Benzi, Pierre Vandergheynst, and Xavier Bresson. FMA: A Dataset for Music Analysis. In *Proceedings of the 18th International Society for Music Information Retrieval Conference, (ISMIR)*, pp. 316–323, 2017.

Finale Doshi-Velez and Been Kim. Towards a Rigorous Science of Interpretable Machine Learning. *arXiv e-prints*, arXiv:1702.08608, 2017.

Alexey Dosovitskiy and Thomas Brox. Inverting Visual Representations with Convolutional Networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*, 2016.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualising Higher-Layer Features of a Deep Network. Technical Report 1341, University of Montreal, June 2009.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of the 33rd International Conference on Machine Learning, (ICML)*, pp. 1050–1059, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NeurIPS)*, pp. 2672–2680, 2014.

Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved Training of Wasserstein GANs. *arXiv e-prints*, arXiv:1704.00028, 2017.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Proceedings of the 30th Annual Conference on Neural Information Processing Systems (NeurIPS)*, pp. 6629–6640, 2017.

R. Devon Hjelm, Athul Paul Jacob, Tong Che, Kyunghyun Cho, and Yoshua Bengio. Boundary-Seeking Generative Adversarial Networks. *arXiv e-prints*, 1702.08431, 2017.

Andreas Krug and Sebastian Stober. Introspection for Convolutional Automatic Speech Recognition. In *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP*, pp. 187–199, 2018.

Kyungyun Lee, Keunwoo Choi, and Juhan Nam. Revisiting Singing Voice Detection: A Quantitative Review and the Future Outlook. In *Proceedings of the 19th International Society for Music Information Retrieval Conference, (ISMIR)*, pp. 506–513, 2018.

Aravindh Mahendran and Andrea Vedaldi. Understanding Deep Image Representations by Inverting Them. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (CVPR)*, pp. 5188–5196, 2015.

Saumitra Mishra, Bob L. Sturm, and Simon Dixon. Undertanding a Deep Machine Listening Model Through Feature Inversion. In *Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR)*, pp. 755–762, 2018.

Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for Interpreting and Understanding Deep Neural Networks. *Digital Signal Processing*, 73:1–15, 2018.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep Neural Networks Are Easily Fooled: High Confidence Predictions for Unrecognizable Images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 427–436, 2015.

Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the Preferred Inputs for Neurons in Neural Networks via Deep Generator Networks. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems (NeurIPS)*, pp. 3387–3395, 2016a.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. *Visualization for Deep Learning workshop, International Conference in Machine Learning*, 2016b. arXiv preprint arXiv:1602.03616.

Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. Plug & Play Generative Networks: Conditional Iterative Generation of Images in Latent Space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature Visualization. *Distill*, 2017.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. *arXiv e-prints*, arXiv:1511.06434, 2015.

Mathieu Ramona, Gaël Richard, and Bertrand David. Vocal Detection in Music with Support Vector Machines. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1885–1888, 2008.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Model Agnostic Interpretability of Machine Learning. In *Proceedings of the ICML Workshop on Human Interpretability in Machine Learning*, 2016a.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016b.

Jan Schlüter and Thomas Grill. Exploring Data Augmentation for Improved Singing Voice Detection with Neural Networks. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR)*, 2015.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *Workshop at International Conference on Learning Representations(ICLR)*, 2014.

Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding Neural Networks Through Deep Visualization. In *Deep Learning Workshop, International Conference on Machine Learning (ICML)*, 2015.

Yichi Zhang and Zhiyao Duan. Visualization and Interpretation of Siamese Style Convolutional Neural Networks for Sound Search by Vocal Imitation. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2406–2410, 2018.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object Detectors Emerge in Deep Scene CNNs. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

Appendix

Table 3: Performance comparison between the model trained by Schlüter & Grill (2015) ("Original") and our replication ("Replication") on the Jamendo test dataset. We can see that both models are very close in their predictive capability.

Model	Threshold	Precision	Recall	Specificity	F1-score	Classification error
Original	0.47	0.901	0.926	0.912	0.913	0.082
Replication	0.50	0.896	0.925	0.908	0.910	0.084

Table 4: The architecture of SVDNet introduced by Schlüter & Grill (2015). Conv, MP and FC refer to the convolutional, max-pooling and fully-connected layers, respectively. Input and output shapes are ordered as: time $\times$ frequency $\times$ number of channels for the Conv layers.

Layer	Input shape	Filter size	Stride	No. of filters/neurons	Output shape
Conv	$115 \times 80 \times 1$	3×3	1×1	64	$113 \times 78 \times 64$
Conv	$113 \times 78 \times 64$	3×3	1×1	32	$111 \times 76 \times 32$
MP	$111 \times 76 \times 32$	3×3	3×3	-	$37 \times 25 \times 32$
Conv	$37 \times 25 \times 32$	3×3	1×1	128	$35 \times 23 \times 128$
Conv	$35 \times 23 \times 128$	3×3	1×1	64	$33 \times 21 \times 64$
MP	$33 \times 21 \times 64$	3×3	3×3	-	$11 \times 7 \times 64$
FC	$11 \times 7 \times 64$	-	-	256	256×1
FC	256	-	-	64	64×1
FC	64	-	-	1	1

Table 5: Best, median and worst hyper-parameter configuration found during hyper-parameter search in section 2.4. Columns indicate name, learning rate, GAN prior weight, number of SGD iterations, and the resulting value of our FID-based evaluation metric, respectively.

Config. name	l_r	λ	N_t	FID
C1	0.01	0.001	100	1.654
C2	0.01	0.01	500	35.880
C3	0.1	0.001	1000	1557.733

Table 6: Reponse for each Mel spectrogram in Figure 4. a_1, a_2, a_3 and a_4 refer to reponse values to the corresponding noise vector.

Category	a_1	a_2	a_3	a_4
$f_g(\tilde{z}_i)$	-0.5	-0.94	1.55	-3.47
Maximisation	5.83	5.52	6.1	2.96
Minimisation	-6.23	-7.21	-6.45	-6.36