Deep Learning Project

MASTER'S DEGREE PROGRAM IN

DATA SCIENCE AND ADVANCED ANALYTICS

A GENERATIVE ADVERSARIAL NETWORK (GAN) FOR THE GENERATION OF DEEPFAKES

Group 22

Alberto Parenti - m20211304

Leonardo De Figueiredo - m20210667

Leonor Candeias – m20210990

Kevin Goodman - m20210701

Ricardo Arraiano – m20210995

04/2022

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

1 Abstract

A generative adversarial neural network (GAN) was made using a training dataset of 9780 facial images and trained in six configurations over 20,000 epochs in a batch-mode format generating fake facial images and attempting their detection. Learning rate, batch size and activation function of hidden layers were altered to determine best model performance. Modest batch sizes of 2-3% of training data with smaller learning rate (0.0001) and Leaky-ReLU activation functions, which avoided the 'dead neuron' problem, outperformed other configurations.

2 Introduction

With the rise of Machine Learning (ML) and neural networks (NNs) has come a time where information can be synthetically made to deceive and mislead. This information can be encoded as images, audio, text or combined into video formats. One outlet of these so called "Deepfakes" is to make a highly influential figurehead appear to be saying something they didn't and thus sway public opinion. Concordantly the need for discrimination on the part of the viewer has risen. This issue has recently resurged with political events involving Russia and Ukraine. The Western media portrays itself as a beacon of truth and virtue, only interested in the common good. Meanwhile the Russian media has likewise labelled itself as a 'liberator' of Ukraine and condemned the expansion of NATO as overly hostile; labelling the United States as hypocritical. Both sides have partially or fully censored the media of the other side. The average citizen is left to themselves to determine which information is real and which is false.

In 2014 Goodfellow et al. [1.] developed a new type of NN whereby two networks – a generator model (G) and a discriminator model (D) – compete against one-another using a training set of real images. The discriminator model works to determine which input image is real (1), and which input image is fake (0); while the generator model tries to "trick" the discriminator model by improving its ability to generate images. This results in a zero-sum game where the two neural networks compete and ideally converge on output images resembling real human faces.

The process proceeds as follows: the discriminator model is built and trained using real images with an input space representing the dimensions of the image by number of pixels and the colour-scale to be used. Each value represents the intensity of colour for that pixel. These

training examples are labelled as unity to refer to the fact that they are real images. Because this is a binary classification problem, the output layer is typically uses a sigmoid activation function where any value greater than 0.5 signals a real image while any value below 0.5 represents a fake image. Initial examples of fake images are generated using the generator model whose first attempt is made by random initialization of weights. This image will be assigned a target value of zero and fed to the discriminator model for training whereupon the weights of the discriminator are updated using back-propagation via the chain rule. The error function of the discriminator takes the form seen in Equation 1. The discriminator's loss function is composed of the sum of both equations. When seeing real images from the training data, the discriminator's loss is defined as –ln(x) but when it is given generated images from the generator its loss is defined by –ln(1-x). Meanwhile the generator never sees the training data and only uses the –ln(x) formula to define its loss function. The ultimate goal of this process being that the generator produces an image where the discriminator predicts a certainty close to 0.5 (unsure of real or fake) as it's unable to minimize its loss any further.

$$\text{fake image loss} = -\ln(1-x); 0 \leq x \leq 1 \tag{1}$$

$$\text{real image loss} = -\ln(x); 0 \leq x \leq 1 \tag{2}$$

1: Equation (1)(2) - Loss functions using binary cross-entropy for G and D models within our GAN network defined between real (1) and fake (0) in image classification.

In this work the authors have attempted to build such a Generative Adversarial Network (GAN) by tuning learning rate, batch size, and activation function, and applied this to facial image data accessed freely online [2.].

3 Methods

Our initial step was to import the necessary Keras libraries and load images as 200 by 200 arrays of pixels with a third dimension designating colour in RGB. The resulting numpy array was an array of arrays where each internal array designated one of the 9780 images in the dataset [2.]. Next we built a generator model, (G). The model contains five layers where an empty tensor defining the latent space is fed to the first dense layer and initialized every epoch. The number of neurons is doubled from 256, to 512, to 1024, finally to 2048 as per recommendations [3.]. Layers utilize the ReLU and Leaky-ReLU activation functions (Equation 2) and are normalized via a batch normalization function with momentum factor 0.8 [3.]. A final *tanh* activation function (Equation 3) is applied to produce an output of 120,000 neurons which is then reshaped into the 120 by 120 by 3 point image.

$$f(x) = max(x); while x \geq 0, else: 0 \tag{3}$$

$$f(x) = 1(x < 0)(a * x) + 1(x \geq 0)(x) \tag{4}$$

2: Equation (3)(4) - The rectified linear activation function (top) and “leaky” rectified linear activation function (bottom) were used in six model configurations. In the latter a small slope alpha is applied when less than zero.

After the generator model we built a discriminator model, (D). Each layer received a flattened input of 120,000 values [3.]. This model was comprised of four dense layers of descending complexity from 1024 neurons, to 512, to 256, and finally to one output neuron indicating class of image as real (1) or fake (0) established by the sigmoid activation function (Equation 4) and defined as the *‘validity’*.

$$T(x) = \frac{exp(x) - exp(-x)}{exp(x) + exp(-x)} \tag{5}$$

3: Equation (5) - The tanh activation function; outputs are defined between [-1,1].

$$S(x) = \frac{1}{1 + exp(-x)} \tag{6}$$

4: Equation (6) - The Sigmoid activation function; outputs are defined between [0, 1].

Subsequently the models are trained using a written training function. The function accepts the image dataset, number of training epochs desired, batch size, and an interval of number of epochs over which to save image results. The training method was as follows: first the discriminator model would be trained on an equally sampled batch of fake and real images. In each epoch the real images would be randomly sampled from the dataset population while the fake images would be generated using the untrained generator model. The discriminator’s loss values over each epoch were then recorded for real and fake images. Next, the generator receives an empty latent space and a target vector of ones and trains at producing real images using the now trained discriminator, all within the same epoch. A function to save the generated images is accessed via an ‘if’ statement every 500 epochs and the entire process was run for 20,000 epochs (Fig. 1). However, in certain configurations where modelled results were inferior, the process was interrupted before the 20,000th epoch to save computing resources.

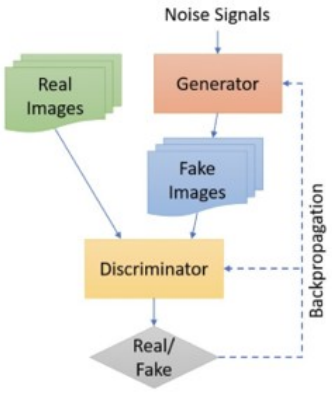

Figure 1: Training mechanism for discriminator and generator models using back propagation. Discriminator accesses both real images from training data and fake images from the generator while the generator uses only its loss function to improve.

The overall combined model (G+D), was run in six configurations (Tab. 1) over twenty-thousand epochs where learning rate, batch size, and activation function were altered to find best results. Model accuracy (number of images assigned correctly out of total number of images) was used as a metric for validating model efficacy throughout while optimization was performed using the 'Adam' algorithm (Equation 5) – a robust variation on stochastic gradient descent[4.].

	Model 1	**Model 2**	**Model 3**
Learning rate	0.0001	0.0001	0.0001
Batch size	128	256	512
Epochs	20000	20000	20000
Loss function	cross entropy	cross entropy	cross entropy
Activation function	LeakyReLU	LeakyReLU	LeakyReLU
	Model 4	**Model 5**	**Model 6**
Learning rate	0.0002	0.0002	0.0002
Batch size	128	256	512
Epochs	20000	20000	20000
Loss function	cross entropy	cross entropy	cross entropy
Activation function	ReLU	ReLU	ReLU

Table 1: Six model configurations performed altering activation function, learning rate, and batch size

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{(E[G_{t,ii}] + \varepsilon)}} * E[g_{t,i}] \tag{7}$$

5: Equation - Adam (adaptive moment estimation) optimization algorithm using gradient mean first moment expectation (E[g]) and variance expectation (E[G]) to attenuate learning rate (η) and update model weights (θ). An exponential decay rate for the first moment (mean) of 0.5 was used for this work with altered learning rates. The second moment decay rate was left at Keras' default value, (0.999).[4.]

Finally a convolutional neural network model (CNN) was attempted over 20,000 epochs. The model built off the same theory as the GAN model with the additional use of the Conv2D layer offered by Keras utilizing a kernel size of three with upsampling in the generator model and a twenty-five percent dropout, kernel size of three, stride value of two, and Leaky-ReLU activation function as recommended for the discriminator model [5.].

4 Results and Discussion

Models were configured and run overnight using cloud computing resources as in Table 1. Results after nine-thousand epochs are shown for each configuration in Figure 2 where models two and three performed best. This was echoed by their accuracy scores of 0.78 and 0.80, after nine-thousand epochs.

Results indicate that a smaller learning rate (η) outperformed the larger learning rate of 0.0002 chosen in models 4, 5 and 6; as well the use of the Leaky-ReLU activation function outperformed the more traditional ReLU activation function. In terms of batch size, the fifth and second model configurations shared a batch size of 256 and gave the best results among its peers when holding other parameters constant. This suggests the optimal overall model should have a lower learning rate, Leaky-ReLU activation function and modest batch size of 2-3% of the training dataset. The advantage of the Leaky-ReLU function is that it allows a small learning parameter (α) during back propagation and so avoids the 'dead neuron' problem of the computationally less expensive ReLU activation function. It's possible this is what led to the over- or under-saturation in models 4-6 versus models 1-3 where, even after several thousand epochs, generator images did not improve as its learning gradient vanished; this process has seemingly been explained empirically for validation [3.]. The use of the tanh activation function in the generator model as opposed to the sigmoid function, (as in the discriminator), has also been described [3.] as it allows for both darker and lighter regions of output images to be treated symmetrically and with a steeper differential gradient during back-propagation.

Figure 2: Generated facial image results for six models' configurations attempted, each over 9000 epochs.

Running in the CNN framework led to an undesirable convergence in images over just a few hundred epochs (Fig. 3); the loss quickly approached zero and accuracy 100%. It was estimated this was due to input size mismatching between generator outputs and discriminator inputs whereby the generator was not capturing the entire nuance of the real images in the dataset.

This led to an easy task for discrimination and a generator model that could not learn. Unfortunately the problem couldn't be pursued further due to a lack of resources in cloud computing capacity and time constraints.

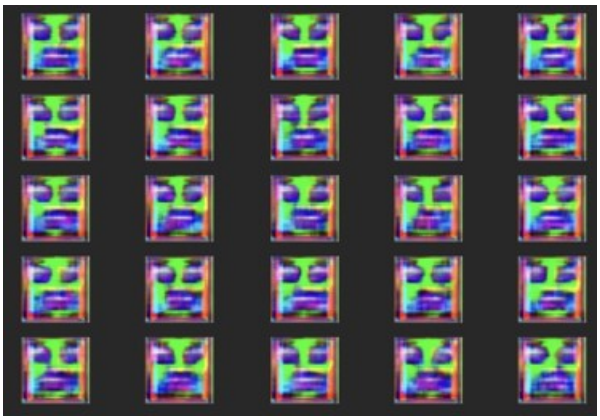

Figure 3: Generator output after training using convolutional neural network layers (CNN).Results converged quickly and training ceased.

5 Future Work

Model improvements could be made by enhanced computer power and time to test a variety of parameter configurations. Ideally the model could be used in a web-interface where the user could upload an image and the result of 'real' or 'fake' could be delivered, although this wasn't attempted due to time constraints. In terms of video data, the model could be improved to strip images from video mp4 inputs and determine if the faces are real or fake, however this would involve more work in handling a variety of image shapes and recognizing when a face was present during the video. The training dataset could also be expanded to include more images with larger variety.

6 References

1. I. Goodfellow et al., *Generative Adversarial Nets*, urly.it/3n424, 2014
2. A. Roy, *Facial Data-Based Deep Learning: Emotion, Age and Gender Prediction*, urly.it/3n443, 2020
3. A. Radford, *Unsupervised representation learning with deep convolutional GAN*, urly.it/3n425, 2016
4. D.P. Kingma, J. Ba, *Adam: A Method for Stochastic Optimization*, urly.it/3n3a6, 2014
5. J. Heaton, *Generating Faces with a GAN in Keras*, urly.it/3n3ap, 2019