Verify It Yourself: A Note on Activation Functions' Influence on Fast DeepFake Detection

Piotr Kawa and Piotr Syga
Wrocław University of Science and Technology, Poland

Keywords: DeepFake Detection, Privacy, MesoNet, Activation Function.

Abstract: DeepFakes are videos that include maliciously added in the postprocessing changes, quite often substituting face of a portrayed individual with a different face using neural networks. Even though the technology gained its popularity as a carrier of jokes and parodies, it raises a serious threat to ones security – via biometric impersonation or besmearing. In this paper we focus on a method that allows of detecting DeepFakes for a user without significant computational power. In particular, we enhance MesoNet (Afchar et al., 2018) by replacing the original activation functions. We achieve over 1% improvement as well as increasing the consistency of the results. Moreover, we introduced and verified a new activation function — Pish that at the cost of slight time overhead allows even higher accuracy on certain datasets.

1 INTRODUCTION

The progress in the field of machine learning provided tools that automatize tasks that previously required high skills, specialized software and were highly time–consuming. One of such tasks is manipulation of facial image — nowadays, thanks to the DeepFake technology it is possible to create face manipulation videos, on a level of detail unachieveable before. DeepFakes require facial images of two persons — the individual present on the source material and the victim whose face will be added to the video. Using these two sets, two networks are trained in order to reconstruct the appropriate face — in result getting expression and illumination of the victim, while keeping facial features of the first person. Recently more sophisticated versions of DeepFakes were introduced using Generative Adversarial Networks (GAN) (Goodfellow et al., 2014). Easy and open access to such technologies leads to many threats that concern various spheres of people's lives. DeepFakes can be used to synthesize besmearing videos, very often of a pornographic nature or depict a public figure saying or doing things that might harm their reputation. Not only celebrities may be affected as shown in (Zakharov et al., 2019), only several images of a person may be sufficient to create such a video. In addition, existence of DeepFake videos in a transparent way undermines the validity of the evidence in courts of law. Naturally, during trials one can be cleared of charged, after detailed examination of the video, yet the process is time–consuming and the trust of general opinion may be irreparably tarnished. In such cases a quick and easy method of legitimacy verification cannot be overstated. Providing methods that allow DeepFake detection without significant computational power and without delay is even more important, due to increase in remote communication, including financial matters.

State–of–the–art DeepFake detection models based on deep neural networks provide results that may be used in courts to examine video evidence. However, their high computational cost makes them unavailable for an individual user. Typically, such scenarios are handled by the use of external services examining the uploaded data, yet highly controversial nature of these videos and costs such external services may not be suitable for everyone wanting to verify the news or clear their name. Notably, other architectures like MesoNet (Afchar et al., 2018), which despite not providing cutting edge results, are good enough to allow an average citizen to have an access to a tool that allows to determine the legitimacy of the material.

Due to the fact that modern DeepFake solutions utilize deep neural networks, one of their main components, that highly influences final performance, is an activation function. By following linear transformation in the neuron, activation function allows neural networks to map non–linear nature of the data.

Kawa, P. and Syga, P.
Verify It Yourself: A Note on Activation Functions' Influence on Fast DeepFake Detection.
DOI: 10.5220/0010581707790784
In *Proceedings of the 18th International Conference on Security and Cryptography (SECRYPT 2021)*, pages 779-784
ISBN: 978-989-758-524-1

One of most widely used activation functions are ReLU and LeakyReLU — both used in MesoNet.

In the paper we focus on detecting DeepFake videos in a fast and reliable manner, so that users may, using their personal PCs, determine if the presented recording is legitimate, before the damage to the reputation spreads. We present a comparison of the performances achieved by MesoNet DeepFake detection model when trained using different activation functions. Moreover, we present and describe a new activation function — Pish. Furthermore, we show, that the initial performance can be increased by using other functions. In addition, we compare the performance of Pish with state–of–the–art solutions using well–known neural networks, e.g., SqueezeNet (Iandola et al., 2016).

2 PREVIOUS WORK

DeepFake detection methods use approaches like frequency analysis (Durall et al., 2019) where authors look for marginal inconsistencies, local feature descriptors (Akhtar and Dasgupta, 2019) or head pose estimation (Yang et al., 2018) exploiting image inconsistencies, when frontal face is embedded into tilted head. Solutions based on deep neural networks become increasingly popular as an approach in DeepFake detection. Their high performance in classification tasks successfully transfers to the problem of detecting facial manipulations. In (Rössler et al., 2019) various methods were evaluated on the dataset — XceptionNet achieved best results. Similarly, the best solution (Seferbekov, 2020) in DFDC Challenge (Dolhansky et al., 2020) was based on the ensemble of EfficientNet networks (Tan and Le, 2019). These methods, despite achieving good results, have high computational requirements which makes them inaccessible for many.

A different approach, has been presented in (Afchar et al., 2018), where a relatively shallow architecture that requires significantly lower resources (28615 parameters in comparison to 23.5 million in ResNet50 (He et al., 2015)), yet at the cost of accuracy resulting in accuracy at the level of 0.873 for low quality images, such as used in our case. More details on MesoNet are presented in Sect. 4. For broader analysis of DeepFake detection methods, their vulnerabilities and the datasets used one can refer to (Nguyen et al., 2020; Lyu, 2020; Tolosana et al., 2020; Yisroel Mirsky, 2020).

In recent years there was also improvement in the field of activation functions. Recently introduced solutions, Swish (Ramachandran et al., 2017) and Mish (Misra, 2019), showed an increase in accuracies when compared to most of the baseline solutions.

3 ACTIVATION FUNCTIONS

Various activation functions were introduced throughout recent years (e.g., (Glorot et al., 2011; Maas et al., 2013; Ramachandran et al., 2017; Misra, 2019)) with some showing progress in relation to their predecessors. However, they have not replaced baseline solutions as their superiority was not universal enough. This has partially changed with the introduction of two activation functions — Swish (Ramachandran et al., 2017) and Mish (Misra, 2019). They were evaluated on many various problems and most of the time outperformed baseline solutions.

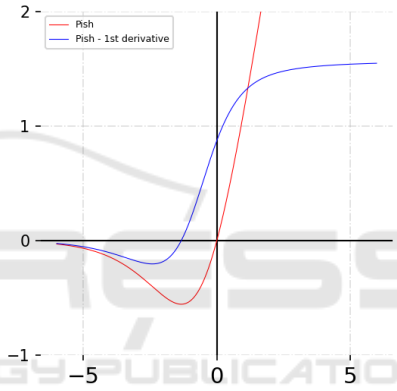

Figure 1: Pish activation function and its derivative.

In this paper we propose Pish (Fig. 1) as a new function, whose properties were inspired by the state of the art functions like Swish and Mish. It is defined by $f(x) = x \cdot \arctan(\text{softplus}(x) + \text{sigmoid}(x))$, where $\text{softplus}(x) = \ln(1+e^x)$ and $\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$. It is a smooth and non–monotonic function, characterized by a shift in values near 0. Values of Pish are bounded from below and unbounded at the top, with a range of values $[\approx -0.56, +\infty)$. Note that the function is continuous, with a smooth derivative $f'(x) = \frac{x\left(\frac{e^{-x}}{(e^{-x}+1)^2} + \frac{e^x}{(1+e^x)}\right)}{\left(\frac{1}{e^{-x}+1} + \log(1+e^x)\right)^2+1} + \tan^{-1}\left(\frac{1}{e^{-x}+1} + \log(1+e^x)\right)$. Its performance has been tested on datasets like MesoNet DeepFake detection dataset, MNIST and CIFAR–10 (Sect. 7). Note that it is slightly more computationally demanding than the aforementioned functions (cf. Sect. 6), yet shows signs of improved performance on some data.

4 MesoNet

MesoNet is an architecture of convolutional neural network proposed in (Afchar et al., 2018) that basing on an input image, determines if presented facial image is a DeepFake manipulation. Method comes in two variants — Meso–4 and MesoInception–4. Meso–4 begins with 4 blocks consisting of a convolutional layer with ReLU, batch normalization and max–pooling layer, followed by a fully–connected layer of 16 neurons with a dropout. MesonInception–4 comes with inception modules (Szegedy et al., 2014) in place of first two blocks.

The solution is shallow in comparison to other architectures like ResNet–50 (He et al., 2015) or XceptionNet (Chollet, 2016) that are commonly used in DeepFake detection. MesoNet solution comes with 27977 and 28615 trainable parameters for respectively Meso–4 and MesoInception–4 variants whereas ResNet–50 is composed of over 23.5 million parameters. Such difference allows using MesoNet in an environment, like the user's PC, that does not provide resources needed for the other architectures and determine the legitimacy of the video in shorter time.

Figure 2: Example of DeepFake image from (Afchar et al., 2018) depicting Jimmy Fallon as Paul Rudd.

5 DeepFake DETECTION EXPERIMENTS

In order to determine the influence of various activation functions, in particular Pish, on the performance of DeepFake detection using MesoNet architecture we performed exhaustive tests. Due to the greater diversity of the model's architecture (use of different convolution kernels in inception module) we have selected MesoInception–4 as a testing subject. It is composed of two activation functions — ReLU, in inception and convolution modules and Leaky ReLU in fully connected layer. Our experiments included ReLU, Leaky ReLU (following the original, using $\alpha = 0.1$), Mish, Swish and Pish. We used same activation functions in both convolution and fully–connected modules.

For a fair comparison we used the same training's and network's parameters whenever they were specified (Afchar et al., 2018). The input data was of size $256 \times 256 \times 3$; the optimization was performed using Adam optimizer. In addition, training process used a batch size of 75 samples and an adjustable learning rate that decreased every 1000 steps by the factor of 10 from 10^{-3} down to 10^{-6}.

Training data underwent augmentation process including zoom, flips, rotations and color appearance parameter adjustments. Augmentation details and number of epochs were selected by us in the course of the experiment, as they were not specified in the original work.

The comparison used the dataset provided by Afchar et al. — 12353 training and 7104 test samples. As no validation set was explicitly separated, it has been extracted by use of 10% of training images. This process was done with the respect to the scenes — in order to prevent data leakage whole scenes were moved to validation set.

We divided the experiments into 3 stages, each of the described processes was repeated several times to ensure that the results are reproducible. We have used all training samples in each epoch which ended with a validation on the whole validation set.

First stage of the experiment concerned training models with a different activation function in both convolution and fully connected modules using all of the aforementioned functions. Each training was performed 12 times for the fixed number of 30 epochs. One training process lasted about 70 minutes using Nvidia RTX 2080Ti, which corresponded to the statement from the original work that few hours of computations on consumer grade GPU were enough to achieve satisfying results.

The second stage aimed to give more details about the efficacy of functions when trained using different learning rates, similar to (Ramachandran et al., 2017; Misra, 2019). We investigated best 3 combinations from the previous stage together with the original one and used two different data splits.

We repeated previous process using different learning rates — functions were trained using decreasing learning rates starting with 10^{-2}, 10^{-3} and 10^{-4}. This resulted in the final rates of correspondingly 10^{-5}, 10^{-6} and 10^{-7}. Each of the 12 training procedures was performed 16 times.

The final stage of the experiment was performed on 5 architectures. They were chosen basing on their

maximum validation accuracy in the second stage (averaged across corresponding experiments). For each of the chosen architectures we have selected 5 models that scored the highest validation accuracy (25 models). Networks were later evaluated on the testing set of 7104 images provided by (Afchar et al., 2018).

In addition to the evaluation of the accuracy, we have run tests to estimate the differences between inference times of various models. To achieve that we measured several times how long did it take for each architecture to evaluate 2000 random images.

6 RESULTS

Results from the first stage (Table 1) clearly showed the increase in the case of other activation functions. Each of the non–original functions provided a better performance that the baseline approach which used ReLU activation.

Table 1: The averaged results of the best accuracy and loss for both training and validation sets scored by the models in the first stage of the experiment.

Activation	Acc.	Loss	V. Acc.	V. Loss
L. ReLU	0.973	**0.027**	**0.923**	**0.059**
Pish	0.954	0.047	**0.923**	0.060
Swish	**0.976**	0.031	0.918	0.064
Mish	0.949	0.049	0.912	0.067
ReLU	0.966	0.035	0.909	0.067

Note that the best performance was achieved by Leaky ReLU and Pish. In order to select the functions for the next stage we have based on the maximum validation accuracy averaged across all the iterations of the process. Basing on this criteria we have selected three activation functions — Leaky ReLU, Pish and Swish along with the the originally used ReLU + Leaky ReLU combination.

Table 2 contains data of 10 best models from the second stage along with their metrics and learning rates used in the training process. The best results were achieved by Swish function with Pish as a close second achieving almost the identical accuracy, whereas Leaky ReLU, the best architecture from the previous stage, performed worse. Top 5 of the presented architectures were selected for the final stage.

Results of the final stage of our experiment are presented in Table 3. Swish function achieved best mean test accuracy with the number of 0.891 while keeping the highest minimum accuracy (at the level of 0.865). Interestingly Pish achieved overall highest test accuracy of 0.928. The original architecture based on ReLU and Leaky ReLU (with the learning rate of 10^{-2}) scored 3rd best results throughout minimum, mean and maximum metrics.

Table 2: Top 10 validation metrics obtained from the second stage of the experiment involving training process with the use of different learning rates.

Activation	LR	V. Acc.	V. Loss
Swish	10^{-3}	**0.905**	0.071
Pish	10^{-3}	**0.905**	0.071
Leaky ReLU	10^{-2}	0.900	0.074
Leaky ReLU	10^{-3}	0.899	0.074
ReLU+L.ReLU	10^{-2}	0.880	0.086
ReLU+L.ReLU	10^{-3}	0.877	0.090
Swish	10^{-2}	0.875	0.092
Pish	10^{-2}	0.844	0.108
Leaky ReLU	10^{-4}	0.829	0.115
ReLU+L.ReLU	10^{-4}	0.829	0.115

Table 3: The results of the final stage of the process involving testing best architectures from the previous stage on the test set (values come from the evaluation of the best 5 models for each of the selected architectures).

Activation	LR	Min	Mean	Max
Swish	10^{-3}	**0.865**	**0.891**	0.918
Leaky ReLU	10^{-3}	0.844	0.888	0.907
ReLU+L.ReLU	10^{-2}	0.793	0.878	0.911
Pish	10^{-3}	0.733	0.857	**0.928**
Leaky ReLU	10^{-2}	0.793	0.848	0.911

Afchar et al. provided pretrained weights of their architectures — these models were also evaluated in the experiment. MesoInception–4 achieved the maximal accuracy of 0.917 on test set. This value differs from the results achieved by the corresponding architecture presented in Table 3 due to differences between training processes. As some of the details like number of epochs or degrees of augmentations were not specified, we could not exactly reproduce original training procedure.

As mentioned, our evaluation process covered not only the performance of activation function but also times of inference. All activations, except for Pish, utilized their official implementations. Treating ReLU as a baseline, we obtained +0.929% for Leaky ReLU, +1.074% for Mish, +2.217% for Swish, and +3.174% for Pish. Note that, since (Afchar et al., 2018) did not provide the exact time requirements of their training and inference, hence we need to treat those as a reference.

Described experiment shows that use of newer activation functions can indeed improve the performance of the neural network. New activation function introduced in this paper — Pish, achieved results that

Table 4: Inference times achieved on 1 element batches of random images. Results averaged across 2000 predictions.

Activation	Inference Time	Relative
ReLU	31.757ms	-
Leaky ReLU	32.052ms	+0.929%
Mish	32.098ms	+1.074%
Swish	32.461ms	+2.217%
Pish	32.765ms	+3.174%

were comparable to state–of–the–art functions. Despite being slightly more demanding in the terms of required computations, the provided results showed that it is a promising and reliably consistent alternative to the existing solutions and that it is worth to explore further its characteristics.

7 ADDITIONAL PISH TESTS

In (Misra, 2019), aside from proposing Mish activation function, extensive tests and benchmarks were provided in the related repository (Misra, 2021). The following section contains the result gathered from some of these benchmarks that were later used to compare Pish with well–known activation functions.

Various Depths of the Neural Network. First test aimed to give information about the performance of model in relation to its depth. The architectures used in the benchmark differed only in the number of fully connected layers. Following author we have used MNIST dataset (Lecun et al., 1998).

Each network started with 2 convolutional layers, followed by a max pooling with a dropout (25%). It was followed by a number of fully connected layers (from range from 15 to 25) of 500 neurons with batch normalization and dropout (25%). Final layer of the network was composed of output neurons using softmax activation function. Both convolutional and fully–connected layers used the currently evaluated activation function. Discussed benchmark was performed using ReLU, Swish, Mish and Pish functions. The utilized optimizer was stochastic gradient descent (SGD), all training processes used the same learning rate with a batch size of 128 and were conducted for 20 epochs.

Figure 3 shows test accuracies of the aforementioned activation functions. Presented results are the maximum test accuracies scored for each consecutive number of layers. Note that, the values remained similar for the smaller number of layers. As their number increased, so had the differences between functions' performance. While the differences between

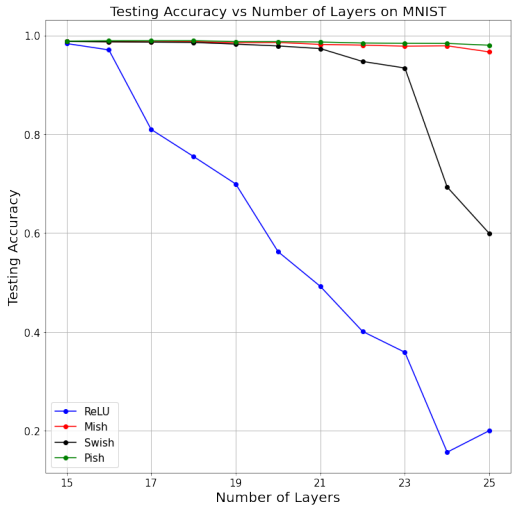

Figure 3: Maximum test accuracy in relation to the depth of the neural network. Plot generated using a script from (Misra, 2021).

Mish and Pish are not big, Pish achieves the top accuracies in the case of almost all network's depths.

SqueezeNet. The second benchmark concerned evaluation of popular architecture SqueezeNet (Iandola et al., 2016). Described process was performed using CIFAR–10 dataset (Krizhevsky, 2012). All activation functions were trained for 100 epochs using the same parameters — batch size of 128 and a learning rate equal of 10^{-3}. Process was repeated 3 times to ensure the reliability of the results.

Table 5: Top–1 test accuracies of SqueezeNet neural network trained on CIFAR–10 dataset.

Activation function	Top–1 accuracy
Swish	**0.887**
Pish	0.886
Mish	0.885
ReLU	0.878

Table 5 contains test accuracies of SqueezeNet trained with ReLU, Mish, Swish and Pish. Models were evaluated after each epoch — presented values are the maximum achieved by each of the networks. The results of the well known functions are as expected. ReLU function provided smallest accuracy, about 0.01 less than the other functions, which is a typical outcome confirmed by many benchmarks. Meanwhile, results of Mish and Swish are close, respectively 0.887 and 0.885. As stated in the original Mish paper, there are cases when Mish function performs slightly worse than the other one. The second best score, 0.886, belonged to Pish.

8 CONCLUSION

In the paper we focus on DeepFake detection, that becomes a serious threat to the privacy and biometric security. We put particular interest in providing methods of detecting DeepFakes feasible for consumer grade devices allowing anyone to verify, with reasonable probability, if the video is legitimate. Our contribution is twofold – we investigated the efficacy of a shallow neural network MesoNet, with various activation functions. Aside experimental verification of previously introduced function, we presented a novel one – Pish, that achieves results competitive to top-performing ones. We achieve over 1% increase over the original solution in both average test accuracy and maximal test accuracy with Swish and Pish activation functions, respectively. Such increase is substantial, as even short DeepFake video (1 minute in length) consists of 1500 images. This allows the user to increase his certainty about the authenticity of a video on his own personal computer.

Moreover, we present an evaluation of performance of Pish under some well known networks. The tests show that the new activation function may be an interesting alternative. Further research in the aspect of activation function should focus on providing optimized implementation of its calculation, so that it generates lower overhead, moreover evaluation of the function on other networks may provide interesting applications resulting in high accuracy.

ACKNOWLEDGEMENTS

This work is partially supported by Polish National Science Centre – project UMO-2018/29/B/ST6/02969.

REFERENCES

Afchar, D., Nozick, V., Yamagishi, J., and Echizen, I. (2018). Mesonet: a compact facial video forgery detection network. *CoRR*, abs/1809.00888.

Akhtar, Z. and Dasgupta, D. (2019). A comparative evaluation of local feature descriptors for deepfakes detection. In *2019 IEEE International Symposium on Technologies for Homeland Security (HST)*, pages 1–5.

Chollet, F. (2016). Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357.

Dolhansky, B., Bitton, J., Pflaum, B., Lu, J., Howes, R., Wang, M., and Ferrer, C. C. (2020). The deepfake detection challenge (dfdc) dataset.

Durall, R., Keuper, M., Pfreundt, F.-J., and Keuper, J. (2019). Unmasking deepfakes with simple features.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. *Proceedings of the 14th International Conference on Artificial Intelligence and Statisitics (AISTATS) 2011*, 15:315–323.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

Iandola, F. N., Moskewicz, M. W., Ashraf, K., Han, S., Dally, W. J., and Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR*, abs/1602.07360.

Krizhevsky, A. (2012). Learning multiple layers of features from tiny images. *University of Toronto*.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lyu, S. (2020). Deepfake detection: Current challenges and next steps.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*.

Misra, D. (2019). Mish: A self regularized non-monotonic neural activation function.

Misra, D. (2021). digantamisra98 mish github repository. https://github.com/digantamisra98/Mish. Accessed: 15-03-2021.

Nguyen, T. T., Nguyen, C. M., Nguyen, D. T., Nguyen, D. T., and Nahavandi, S. (2020). Deep learning for deepfakes creation and detection: A survey.

Ramachandran, P., Zoph, B., and Le, Q. V. (2017). Searching for activation functions.

Rössler, A., Cozzolino, D., Verdoliva, L., Riess, C., Thies, J., and Nießner, M. (2019). FaceForensics++: Learning to detect manipulated facial images. In *International Conference on Computer Vision (ICCV)*.

Seferbekov, S. (2020). Kaggle dfdc best solution. https://www.kaggle.com/c/deepfake-detection-challenge/discussion/145721. Accessed: 08-03-2021.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014). Going deeper with convolutions. *CoRR*, abs/1409.4842.

Tan, M. and Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks.

Tolosana, R., Vera-Rodriguez, R., Fierrez, J., Morales, A., and Ortega-Garcia, J. (2020). Deepfakes and beyond: A survey of face manipulation and fake detection.

Yang, X., Li, Y., and Lyu, S. (2018). Exposing deep fakes using inconsistent head poses. *CoRR*, abs/1811.00661.

Yisroel Mirsky, W. L. (2020). The creation and detection of deepfakes: A survey.

Zakharov, E., Shysheya, A., Burkov, E., and Lempitsky, V. (2019). Few-shot adversarial learning of realistic neural talking head models.