UTILIZATION OF AUTO ENCODERS AS A GENERALIZATION METHOD FOR ANTI-SPOOFING CLASSIFICATION

Er.Vishal Kumar[1], Ms.Rachna Rajput[2]
[1,2]Guru Kashi University, Talwandi Sabo

ABSTRACT

When training large neural networks, explicit generalisation procedures are required to ensure that the model's predictions can be generalised. Auto-encoders now have a new generalisation result. As a result of the data's limited diversity, generalisation issues arise when developing models aimed at tackling these types of problems. Regularization is often added by including a regularisation parameter in the objective function; however, in this paper, we will provide an alternative regularisation approach that is equal to the one commonly used. The fact that other regularisation strategies failed to eliminate overfitting implies that these regularisation techniques may be used as a pre-processing step for other algorithms.

KEYWORDS: Regularization, Algorithm, Auto encoders, Generalization.

I. INTRODUCTION

Not only can a neural network learn data characteristics via convolutions, but it may also employ other methods as well. When an autoencoder is trained, it tries to learn an identity function representation that produces an output that matches the input. The encoder, f(x), and decoder, g(f(x), are the two pieces of these networks. We're more interested in the first section. The sole difference between an autoencoder and a feedforward neural network is the eventual goal (different loss function). Restrictions may be put in place to prevent overfitting and learning the identity function, splitting autoencoders into undercomplete and regularized subcategories. A spoof is an attempt to mimic another user in order to obtain access to data, transmit malware, or engage in any other malicious cyber-attack. Facial spoofing techniques will be discussed in depth, since most spoofing assaults these days target system protocols and addresses. The most basic approach is to show a photo, which is quite easy to do due to the prevalence of images of faces on social media.

II. LITERATURE REVIEW

Romain Cosentino et al (2021) An important unsolved problem in deep learning is how approaches generalize when there are more model parameters than instances to train on. One of the most widely used deep learning techniques, Deep Autoencoders (AEs), is the subject of our research. We want to shed light on the underlying phenomena of this technique. In particular, we use the continuous piecewise affine structure of AEs to explain how they resemble the data manifold. New insights on their mapping, reconstruction guarantees, and an interpretation of frequently used regularization approaches are provided by our reformulation of AEs. We use these discoveries to develop two novel regularizations that

allow AEs to capture the data's intrinsic symmetry. With our regularizations, AEs may better approximate the data manifold without having to define the group underlying it explicitly, thanks to recent breakthroughs in the group of transformation learning. We show that the regularizations guarantee the generalization of the related AEs if the symmetry of the data can be described by a Lie group.

Solinas et al (2020) Reinjections may be used to transform non-generative autoencoders (such as contractive or denoising autoencoders) into generative models, which is what generative autoencoders are supposed to do (i.e., iterative sampling). We show mathematically that any autoencoder that reproduces the input data with a loss of information may sample from the training distribution using reinjections in this paper. To be more exact, we show that the property of modelling and sampling from a given distribution applies to all lossy autoencoders, not only contractive and denoising ones. We underline that reinjection sampling in autoencoders enhances sampling quality in agreement with prior findings. By using non-generative autoencoders trained on standard datasets, we can demonstrate this characteristic.

Karim et al (2018) Many images classification and identification tasks, such as handwriting recognition, medical imaging, and face recognition, have been solved using deep autoencoder neural networks. When building deep autoencoder neural networks, the number of parameters, network topology, and transfer function compatibility all have a role in total performance. It's possible that an incorrect structure design might have a negative impact on deep autoencoder neural networks' performance. When integrating the Taguchi Method into an autoencoder-based system, the general topology of the network does not need to be altered. Data from diverse domains, such as network security and medical, is used to conduct a number of studies. Although several well-known strategies in the literature performed better, our strategy consistently outperformed those ways. As a consequence, the findings are positive and demonstrate the overall effectiveness of the suggested framework.

Lei Le (2018) When training large neural networks, explicit generalization procedures are required to ensure that the model's predictions can be generalized. A rising number of people are interested in pursuing this aim by using various, sometimes unrelated, jobs. The supervised auto-encoder, a neural network that simultaneously predicts targets and inputs, is the model we theoretically and experimentally examine in this paper (reconstruction). Auto-encoders may be improved by including the reconstruction error into the regularization process—especially in comparison to simple regularization methods such as norms. We next show experimentally that the supervised auto-encoder, when compared to the equivalent conventional neural network, never degrades performance and can increase generalization across a variety of designs with various numbers of hidden units and activation functions.

III. RESEARCH METHODOLOGY

The NUAA Photo Imposter Database is the dataset utilised in this study. It has 12614 photos, 5105 of which are of actual individuals and 7509 of which are photographs of the same persons.

Using a webcam, participants were requested to gaze directly at the camera, mimicking an image as closely as possible (no blinks, no head movements, etc.) in order to ensure this variance. The final dataset was compiled from a selection of frames from these films. We utilise the face detector provided in the OpenCV package, haarcascadefrontalface default.xml, and run it across the dataset, removing certain examples where the detection was unsuccessful. When it came to printing the photo cases, a high-definition camera was used and photographic paper was used. The camera was then used to move the paper around and fold it in various orientations

.

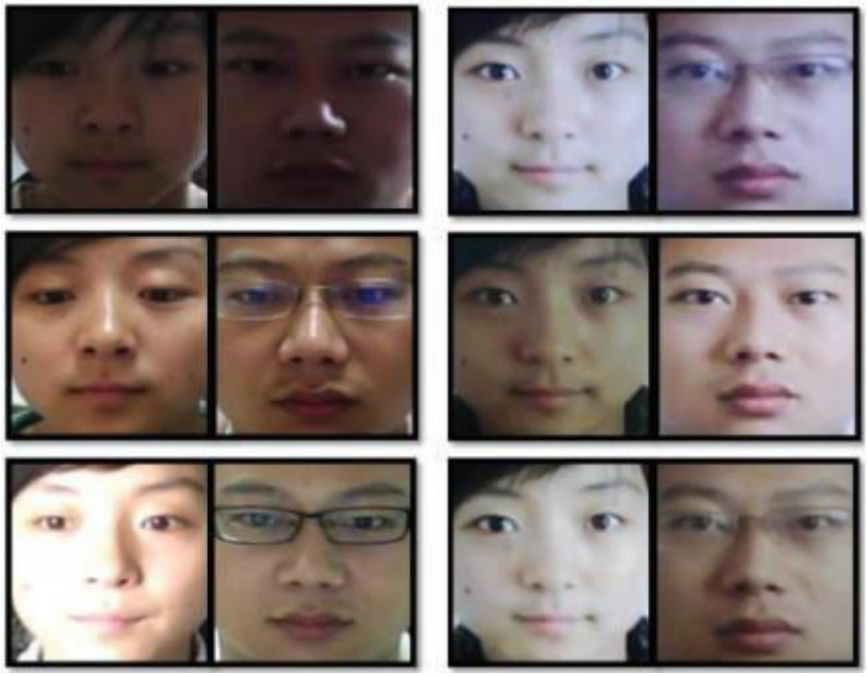

Figure 1

We opted to redistribute the previously formed sets as follows since there was no validation set and the distribution of data for train-test-validation is typically 60-20-20: Validate your new training set using the train set, and then split the test set into two parts: one for the test set, and one for the new training set (still conserving the non- overlapping property).

As soon as we've picked and rearranged the data, we may begin face detection at the same time

	Train	Validation	Test
Real	2200	1180	950
Spoofing	4000	1850	1580
TOTAL	6200	3030	2530

IV. DATA ANALYSIS

4.1 CNN

We can begin building our anti-spoofing neural network as soon as the data is ready. Initialization and pre-training weights will not be used in this CNN. Convolutional and max pooling layers combine in this network's design, which concludes with a flattening and dense layer. The Keras API provides a brief overview. To determine when to cease training, an early stopping point is selected by comparing the training and validation loss with a patience of 20 epochs.

The first result we receive is this:

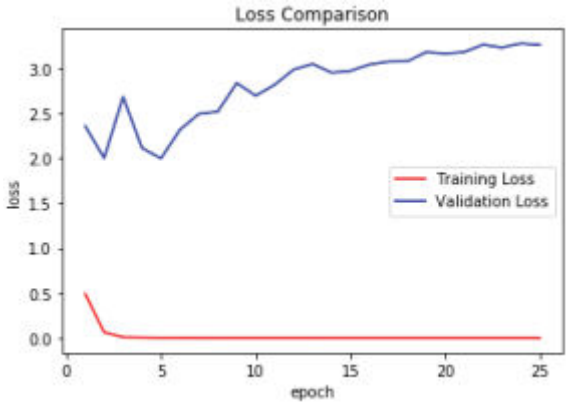

Figure 2

Overfitting is evident in the early epochs, as the training loss rapidly approaches 0 but the validation loss rapidly increases. The most common ways to prevent or at least minimise overfitting are as follows:

Adding new information. Due to the fact that we are dealing with a dataset that has already been created, this isn't an option. Use a different kind of architecture or lower its complexity. For our experiment, we can't modify the architecture's type since it is the only use case, we are interested in. To discover a proper architecture for the complexity, various attempts were made, and the best of them was chosen.

Enhance your data using third-party sources. In our situation, this is relevant. By flipping the training set vertically and altering the shear, range and brightness of the pictures we will experiment with data augmentation techniques.

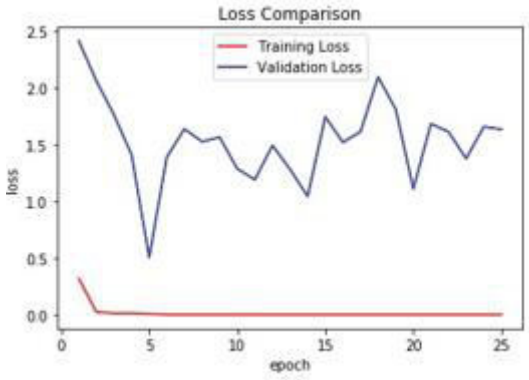

Figure 3

While we are still overfitting, the validation loss does not suddenly start growing in this example, indicating that we may be on the right track.

Last but not least, as a regularisation strategy, we will test dropout.

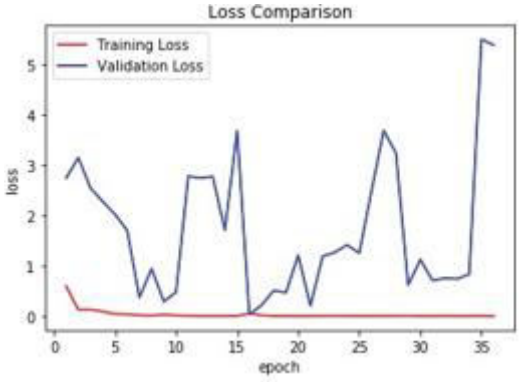

Figure 4

The difference between the two losses has narrowed to the point that it's virtually unnoticeable; nevertheless, while the training loss decreases steadily, the validation loss fluctuates considerably.

4.2 Autoencoder

It's best to take our previous net's convolutional layer (without the flattening and dense layer) and add a symmetrical network to it, swapping out the pooling layers for up sampling ones in order to create an under-complete AE. The AE will be constructed layer-by-layer. To do this, we'll create four tiny AEs, train them independently, and then import their weights back into the original CNN.

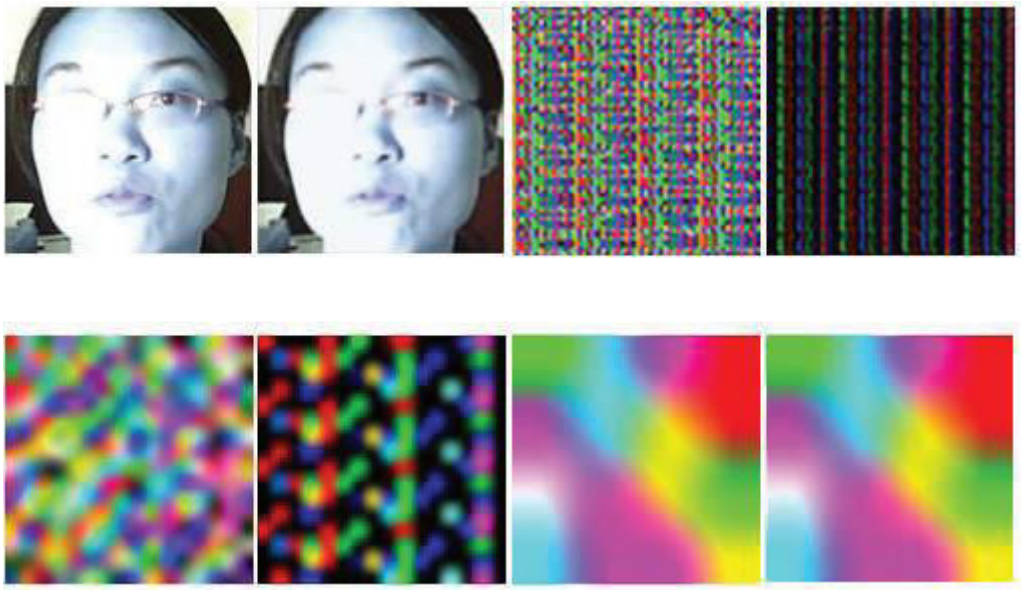

Figure 5

As we go further into the network, we may see a variety of distinct behaviours:

- Using a face as an input, the first autoencoder generates a somewhat poorer quality rendition of the face than the original.
- The second and third autoencoders use the final code from the previous encoder as an input and produce some type of a denoised version of it.
- Autoencoder number four accepts as an input a final code generated by number three, and provides a nearly identical output (at least graphically; if we do a numerical comparison we can observe how input and output are not exactly the same).

```
array([[0., 1., 0., ..., 1., 1., 1.],
       [0., 1., 0., ..., 1., 1., 1.],
       [0., 1., 0., ..., 1., 1., 1.],
       [0., 1., 0., ..., 1., 1., 1.]], dtype=float32)

array([[-1.        ,  1.        , -0.99989986, ...,  1.        ,
         1.        ,  1.        ],
       [-1.        ,  1.        , -0.9999921 , ...,  1.        ,
         1.        ,  1.        ],
       [-1.        ,  1.        , -0.9999942 , ...,  1.        ,
         1.        ,  1.        ],
       [-1.        ,  1.        , -0.9943713 , ...,  1.        ,
         1.        ,  1.        ]], dtype=float32)
```

V. RESULTS

On the last figure, we can observe how our results compare to those of other regularisation methods. Overfitting is no longer a problem with the model. Losses in both training and validation have a similar pattern. There are occasional discrepancies between training and validation losses, but these discrepancies are explained by the use of mini-batches and are not comparable to earlier findings. Despite the increased training loss, it is desirable to lose some accuracy in the training set if, in return, our model performs better on unseen data. Even though no autoencoders show any signs of overfitting, this suggests the combination of non-overfitting solutions at the local level approximates a solid non-overfitting solution at the global level.

We receive a test accuracy estimate of 65 percent when we crunch the numbers. A high-accuracy anti-spoofing system was not the purpose of this study, but rather a system that can generalise effectively on unseen data. The data may also be too similar even if we design a test set that is as dissimilar from the training set as feasible.

VI. CONCLUSIONS

There is a lot of research going on in the subject of computer vision, including picture categorization and object identification. We've also covered what autoencoders are, how they function, and how they're useful for generalisation. If we wish to create a network from scratch, this technique might be beneficial as a weight initialization for those scenarios in which we employ transfer learning and existing deep networks. Data scientists have sought to mitigate spoofing attacks by using a layer-wise building strategy, which we have studied and put into practise. To summarise, we believe that autoencoders should not be employed on their own because of the poor accuracy that they provide. This regularisation approach may be used as a pre-processing step for other algorithms due to its success in eliminating overfitting.

In order to develop an anti-spoofing mechanism, CNN's final model has been preserved.

REFERENCES

1. Romain Cosentino et al (2021),” Deep Autoencoders: From Understanding to Generalization Guarantees”, Proceedings of Machine Learning Research vol 107:1–26

2. Solinas, M. &Galiez, Clovis &Cohendet, R. &Rousset, Stéphane &Reyboz, M. &Mermillod, Martial. (2020). Generalization of iterative sampling in autoencoders. 877-882. 10.1109/ICMLA51294.2020.00143.

3. Karim, Ahmad &Guzel, Mehmet &Tolun, Mehmet & Kaya, Hilal&Celebi, Fatih. (2018). A New Generalized Deep Learning Framework Combining Sparse Autoencoder and Taguchi Method for Novel Data Classification and Processing. Mathematical Problems in Engineering. 2018. 1-13. 10.1155/2018/3145947.

4. Lei Le (2018),” Supervised autoencoders: Improving generalization performance with unsupervised regularizers”,

5. K. Simonyan, A. Zisserman. “Very Deep Convolutional Networks for Large- Scale Image Recognition”, 2014

6. Kollreider et al. “Real-Time Face Detection and Motion Analysis with Application in Liveness Assessment”, 2007

7. Krizhevsky et al. “ImageNet Classification with Deep Convolutional Neural Networks”, 2012. Neural Information Processing Systems Conference.

8. L. Sánchez. “Local Binary Patterns Applied to Face Detection and Recognition”, 2010. UniversitatPolitècnica de Catalunya.

9. Li et al. “Live Face Detection Based on the Analysis of Fourier Spectra”, 2004. Chinese Academy of Science, Beijing

10. Liu et al. “Learning Deep Models for Face Anti-Spoofing: Binary or Auxiliary Supervision”, 2018. Michigan State University.

11. M. Ranzato, Y. Bourear, Y. LeCun. “Sparse Feature Learning for Deep Belief Networks”, 2008. New York University