REGULARIZATION TECHNIQUES IN MACHINE LEARNING

Course Project for IISC CCE Course

Table of Contents

Section 1: A Brief Introduction to Regularization

Overfitting is a common issue faced by machine learning practitioners. Overfitting is a phenomenon where the model fits the training data too well, and the machine learning model ends up modelling the inherent noise in the dataset. This degrades its performance on real world unseen data. Overfitting can also happen when train and test datasets come from different distributions. There are several solutions to overfitting, some of the common ones being increasing training data to allow the model to learn across more samples and loss function regularization, which allows for penalties to be imposed along with the training loss to discourage the model parameters from overfitting the training data. In addition to this

Dropout regularization is another technique, where a random subset of nodes in a given layer are disabled at every mini batch, thereby enabling every node to capture patterns from the training data. Data augmentation is considered to be a weak regularizer, typically because it involves cropping, reshaping and other transformations on existing data. Recent innovations in this space include GANs, Adversarial Auto Encoders, which are considered to be adversarial models where typically a generator and discriminator compete against each other and thereby improving quality of generated data.

In this project, the concept of weight tying is discussed as a regularization technique. In addition to this, in order to increase training data, the concept of adversarial autoencoders is discussed, where the encoder learns a latent space mapping of the training data with some restrictions imposed on the encoder's feature space, which is controlled by a discriminator. Both weight tying and data augmentation are discussed in an autoencoder context, with the Section 4 containing implementation details.

Section 2: Weight Tying Regularization

To illustrate the concept of weight tying, consider the following image in Fig 1.

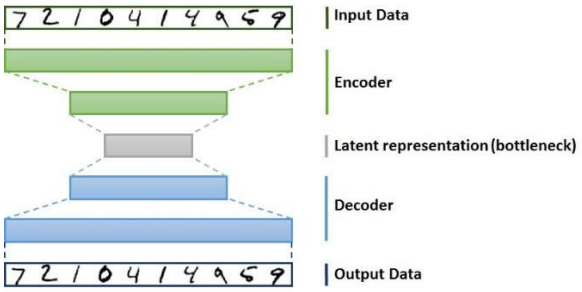

Fig 1: Autoencoder architecture

An autoencoder consists of an encoder, which maps the input to a latent space, and a decoder which remaps it into its original size. The objective of the autoencoder is to minimize the reconstruction loss, thereby ensuring the latent space representation is a fairly accurate representation of the actual image, in reduced dimensions. By tying the weights and making the decoder weights to be a transpose of the encoder weights, we effectively reduce the number of parameters in the model to ½ of its original size, which acts as a natural regularizer. By tying the weights, effectively the model is trying to learn a symmetric mapping from the original image to the latent space and vice versa. Weight tying's advantages include reduced training speed in addition to reduced risk of overfitting.

Section 3: Data Augmentation

Data augmentation is the technique of increasing training samples in order to improve a machine learning model's ability to generalize across a wide variety of samples and thereby perform better on real world data. Some augmentation methods include rotating, cropping, flipping, adding noise, dilation, etc. More recently, Generative Adversarial Networks and AAEs have proven successful at generating synthetic data that are more reflective of the real world and can be used as training data for machine learning classifiers.

3a. Interpolation

Autoencoders learn latent space representations for their input images. To use this for data augmentation, one of the simplest methods to do so is interpolation. Using interpolation, we can have a convex combination of two or more latent space mappings, which is subsequently fed to a decoder to generate a new image. For the MNIST dataset (which is further explained in the implementation section) this has been found to be quite effective at generating high quality synthetic data.

3b. Adversarial Auto Encoders

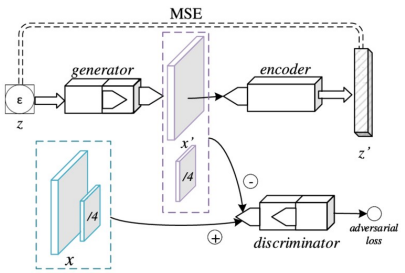

Fig 2: Adversarial Autoencoders architecture

An adversarial autoencoder consists of an encoder and decoder like any autoencoder, however in addition it also contains a discriminator which takes the encoded representations as input. The discriminator, like in GANs has two labels, real and fake. The latent representations generated by the autoencoder are labelled as fake, and sample values from a gaussian distribution (or any other distribution which is desired) is labelled as real and fed to the discriminator. The objective is to make the encoder generate latent space representations in a controlled setting and make the latent representations follow the distribution that is required by the generator. Controlling the distribution also allows for easier interpolation. The distribution function can be set in a way that can reflect the real-world distribution of the data.

Section 4: Implementation

For reconstruction errors, three different autoencoder models are tried out. All of the models have symmetric shapes with respect to encoder and decoder. The three variants include a vanilla autoencoder, an autoencoder with weight tying and an adversarial autoencoder. The weight tying autoencoder is observed to perform better with the SELU activation function, while the vanilla variant uses RELU activations. Each model was trained for 20 epochs on the MNIST handwritten digits dataset. The optimizer used is Adam Optimizer. The reconstruction loss function used is Mean Squared Error (MSE) loss and the loss function used by the discriminator in the AAE model is Binary Cross Entropy Loss. The latent space dimension of the encoder is 8 in all the three models. The encoder uses convolutional layers to extract feature representations while the decoder uses transposed convolutions for reconstructing the images. In the tied weights model, the same kernel used for encoder is used in transposed convolutions thus effectively ensuring weight tying.

Table 1 shows the reconstruction losses obtained on training and test data respectively.

Table 1:

Model	Reconstruction Loss (Train)	Reconstruction Loss (Test)
Vanilla Autoencoder	0.01691859	0.01723348
Tied Weights Autoencoder	0.01638342	0.01685843
Adversarial Autoencoder	0.02728836	0.02037433

We observe that, although marginally the tied weights autoencoder performs best, despite the fact that, it contains just about half the number of parameters, therefore demonstrating the generalizing ability and effectiveness of the weight tying approach.

Table 2 shows the number of parameters for each model and average training time across 20 mini batches on all 3 models.

Table 2:

Model	Time Taken (in seconds)	Model Parameters
Vanilla Autoencoder	11.59391524	88169
Tied Weights Autoencoder	11.16286944	43987
Adversarial Autoencoder	13.37933944	143986

Although the number of parameters in the tied weights model is less than half of the vanilla model and almost 1/3rd of the AAE model, the training times do not have much disparity. The similar performance is attributed due to the fact that although the number of parameters is different, the number of floating-point operations performed have much less differences, especially when comparing the vanilla and the tied weights model. Although the tied weights model has no new parameters for its decoder, it still has to compute the forward pass in the decoder layer, using the transpose of the encoder parameter matrices. This explains the minor reduction in training time for the tied weights autoencoder.

Figures 3a, 3b, 3c (left to right) show plots of training losses for all the three models

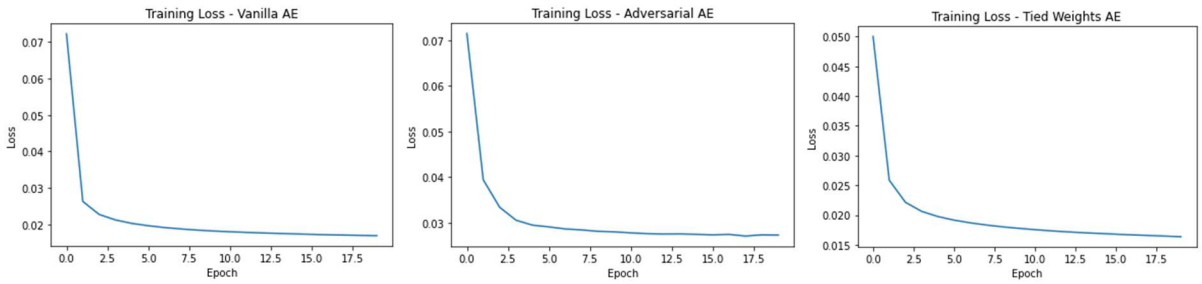

Fig. 3a, Fig. 3b, Fig. 3c

The subsequent section illustrates visually a sample from each autoencoder on interpolation of training data. The interpolation algorithm involves taking the latent space representation of two random images and recombining them in different ratios from 0.1 to 1, in steps of 0.1. Following this, the interpolated latent space representation is fed into the respective decoders to generate the image. The image for each ratio from 0.1 to 0.9 is plotted, along with the original two source images. Fig 4a, 4b, 4c show the interpolation quality of the three models

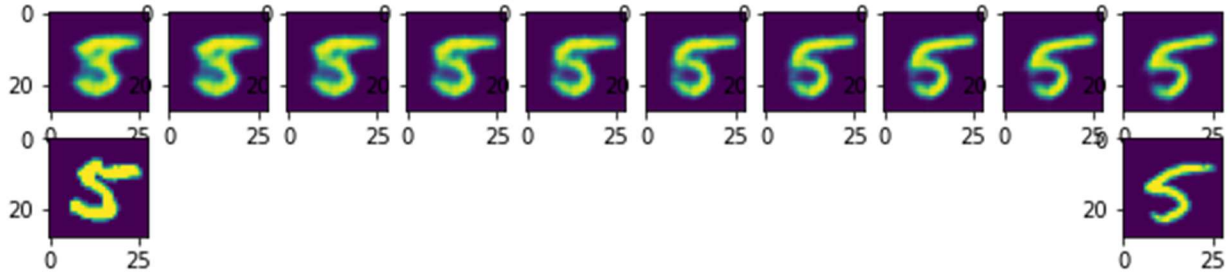

Fig 4a: Interpolation results for Vanilla Autoencoder model

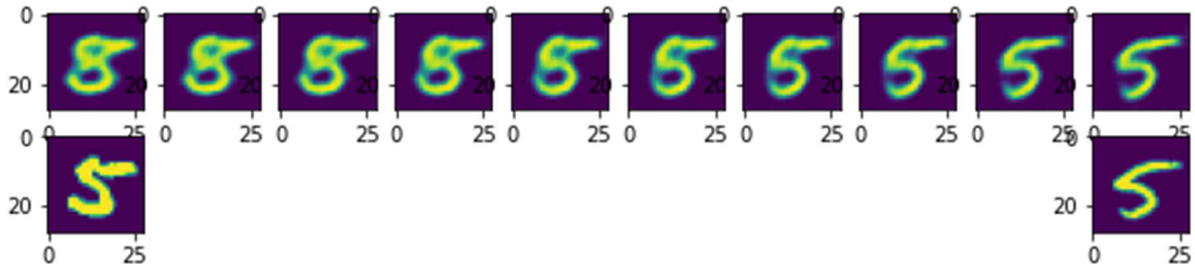

Fig 4b: Interpolation results for Tied Weights Autoencoder model

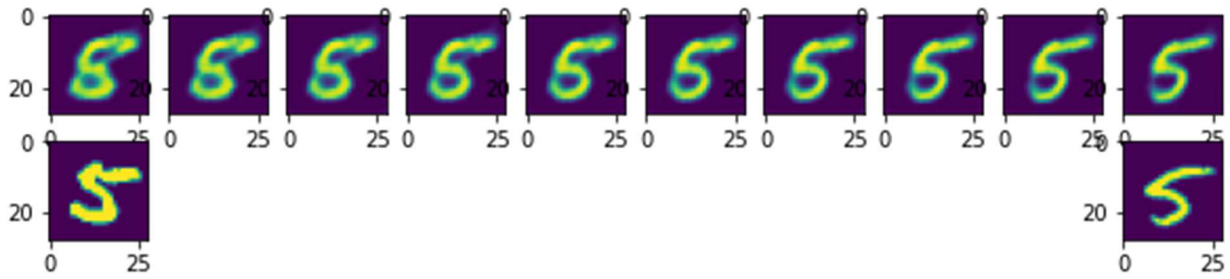

Fig 4c: Interpolation results for Adversarial Autoencoder model

The image quality of interpolations seems the best for the Adversarial model. This could be attributed due to the fact that the distribution of encoder's latent space values is controlled by the discriminator and the encoder of the AAE model does a better job during interpolation. The initial few images for both the tied weight and vanilla models do not seem of good quality. However, all the three models seem effective particularly in the latter half of images generated, thus demonstrating their effectiveness as a data augmentation tool.

Summary:

Three different types of autoencoder architectures were implemented and their performance evaluated with respect to two different regularization techniques – data augmentation and weight tying. For data augmentation, image interpolation's effectiveness was explored. In addition, an adversarial autoencoder model based on the principles of GAN training was trained, and its performance was evaluated on the MNIST dataset.

Conclusion:

This report was drafted and prepared by Rahul Seetharaman (rahul.ml.engg@gmail.com). Sincere thanks to Prof. Kunal Narayan Chaudhury for the CCE Course on Linear Algebra and Optimization for Machine Learning.