Anomaly Detection with Auto-encoder

5LSH0, Computer Vision and 3D Image Processing - Q2 (2023)

Full Name	Student ID
Yokesh Dhanabal	2011468

Eindhoven, February 4, 2024

Contents

1 | Implementation

Implement simple AE architecture in PyTorch by using linear layers (without convolutional layers).
Refer **section 1** in AutoEncoder.ipynb notebook
Section 1.1 - Encoder and Decoder parts
Section 1.2 - Mean Squared Error (MSE) loss function
Section 1.3 - Stochastic Gradient Descent (SGD)

2 | Questions

Question 1 Gaussian Noise with a mean (default value: 0) and standard deviation (default: 0.5) is added to the test images. Also, the Gaussian noise to be added to the images can be controlled by changing the *mean* and *std* variables in the notebook.

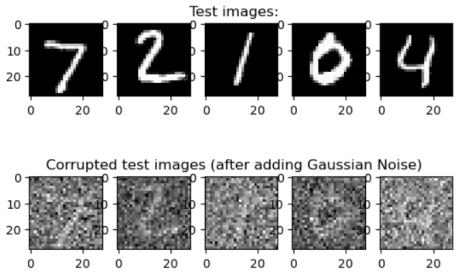

Figure 2.1: Original (Test) images and corrupted (Noisy) images

Question 2

- Results: The quality of the reconstructed image decreases after incorporating noise to original images as seen in.
- Expectation and observation: The auto-encoder produced the expected digit as present in the input, even in the presence of noise. But if we increase (noise) by varying the mean or standard deviation of the Gaussian, then the auto-encoder fails to reconstruct the required digit.
- Reason: The above behaviour is because, in presence of high amount of noise, the network may focus on the noise rather than detecting the required features. Thus, the auto-encoder performs in a poor way to unseen test images that contain noise.

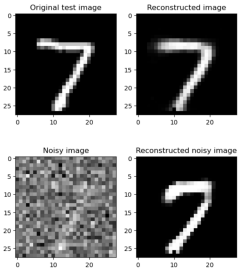

Figure 2.2: Reconstructed images

Question 3
A new data set called 'noisy_dataset' is created and it contains normal training images (50%) and noisy images (50%). Now, when the model the model is trained using noidy data set, the MSE loss is high during the training phase, but the model is generalized to handle noisy images compared to previous case, where the model is trained only using normal images. This is evident by comparing the MSE losses of test set for both cases. MSE loss for test set using the model trained only normal images is 0.6602. Whereas, the model trained using noisy images has a test set loss of 0.6555. Thus, the second model trained using 'noisy_dataset' performs well in test set.

Question 4
SSIM is a metric to measure the similarity between two given images based on the structural information. Common image quality metrics **quantify only the difference in the values of each of the corresponding pixels between the sample and the reference images**. For example, in Mean Squared Error (MSE), the average squared difference between the value observed (pixel value of reconstructed image) and the values predicted by a model (corresponding pixel value of original image).
But, if we consider human visual perception system as an example, which is highly capable of identifying structural information from a scene and finding out the differences between a reference and a sample

scene, a metric was introduced in the 2004 IEEE paper, Image Quality Assessment: From Error Visibility to Structural Similarity.
This provides an intuition to the Structural Similarity Index (SSIM) metric which extracts 3 key features such as **Luminance**, **Contrast** and **Structure** from an image and the comparison between the images is based on the extracted metrics.
This metric calculates the Structural Similarity Index between 2 given images and provides a value between -1 and +1. A value of +1 indicates that the 2 given images are very similar or the same while a value of -1 indicates the 2 given images are very different. In our case, these values are adjusted to be in the range [0, 1], where the extremes hold the same meaning.
Firstly, the Luminance(μ), Contrast(σ) and Structure of individual images are found. Following this, comparison functions that can compare the two images on these parameters are defined and finally a combination function that combines all the above parameters yields the Structural Similarity Index value. After considering above derivations and assumptions, the final expression for SSIM is obtained as given below [1].

$$SSIM = \frac{(2 \cdot \mu_x \mu_y + C_1) \cdot (2 \cdot \sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1) \cdot (\sigma_x^2 + \sigma_y^2 + C_2)}$$

As we are calculating the loss using SSIM, the function($ssim\_loss()$) calculates the SSIM metric for two images/ array of images and outputs a value (SSIM loss) which equals to $1 - SSIM$. Also, numpy functions are used for calculating the Luminance and Contrast, as calculation of mean and standard deviation for each images using manual loops leads to high amount of training time.

Question 5

As an improvement to existing Linear model, a complex model, an Over-complete auto-encoder is implemented and the main difference is that, an Over-complete AE employs a higher number of hidden units than input dimensions. The idea behind using more hidden units is to learn a more complex, non-linear function that can better capture the structure of the data. Also, they are more robust to noise and can handle missing data better than traditional auto-encoders.
We could not observe any distinct difference among the reconstructed images of auto-encoders (Verify AutoEncoder.ipynb notebook for the images), but the complex auto-encoder has low MSE loss on test set compared to previously implemented linear auto-encoder.

Architecture of Auto-Encoder

Layer (type)	Output Shape	Param #
Linear-1	[-1, 1, 256]	200,960
ReLU-2	[-1, 1, 256]	0
Linear-3	[-1, 1, 64]	16,448
ReLU-4	[-1, 1, 64]	0
Linear-5	[-1, 1, 8]	520
Linear-6	[-1, 1, 64]	576
ReLU-7	[-1, 1, 64]	0
Linear-8	[-1, 1, 256]	16,640
ReLU-9	[-1, 1, 256]	0
Linear-10	[-1, 1, 784]	201,488
Sigmoid-11	[-1, 1, 784]	0

Total params: 436,632
Trainable params: 436,632
Non-trainable params: 0

Input size (MB): 0.00
Forward/backward pass size (MB): 0.02
Params size (MB): 1.67
Estimated Total Size (MB): 1.69

Architecture of Complex Auto-Encoder

Layer (type)	Output Shape	Param #
Linear-1	[-1, 1, 1000]	785,000
ReLU-2	[-1, 1, 1000]	0
Linear-3	[-1, 1, 1000]	1,001,000
ReLU-4	[-1, 1, 1000]	0
Linear-5	[-1, 1, 1000]	1,001,000
ReLU-6	[-1, 1, 1000]	0
Linear-7	[-1, 1, 1000]	1,001,000
Linear-8	[-1, 1, 1000]	1,001,000
ReLU-9	[-1, 1, 1000]	0
Linear-10	[-1, 1, 1000]	1,001,000
ReLU-11	[-1, 1, 1000]	0
Linear-12	[-1, 1, 1000]	1,001,000
ReLU-13	[-1, 1, 1000]	0
Linear-14	[-1, 1, 784]	784,784
Sigmoid-15	[-1, 1, 784]	0

Total params: 7,575,784
Trainable params: 7,575,784
Non-trainable params: 0

Input size (MB): 0.00
Forward/backward pass size (MB): 0.11
Params size (MB): 28.90
Estimated Total Size (MB): 29.01

Figure 2.3: Model summary of Linear auto-encoder (left) and Complex (Over-complete) auto-encoder(right)

Question 6

The model summary of the Convolutional auto-encoder and the reconstructed test images can be seen in figure 2.4. The Convolutional auto-encoder has three convolutional layers and three deconvolutional (transposed convolution) layers as seen in the model summary.
Comparing with previous two models, the convolutional auto-encoder shows a better performance. The similarity between the reconstructed image and original image is very high and it also has the lowest loss in test set.

Question 7

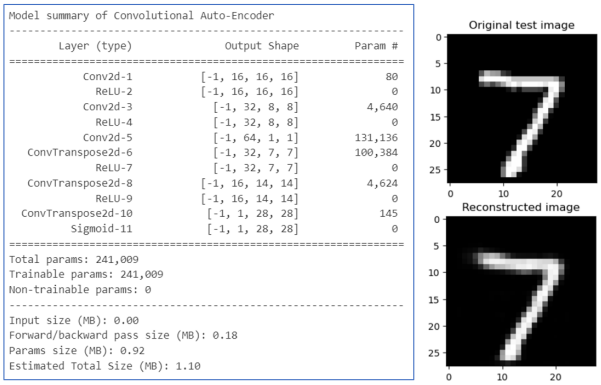

Figure 2.4: Model summary of Convolutional auto-encoder (left) and reconstructed test image (right)

Selecting best model: Based on visual analysis and MSE loss on test set (Simple Linear auto-encoder ≈ 0.65, Complex model ≈ 0.011, Convolutional model ≈ 0.004. Please refer the notebook for exact values and these values may differ for each run). Thus, it is evident that the convolutional auto-encoders provide better results. (Since we are not doing any prediction or classification, MSE loss can be considered as alternative to accuracy metric)
Convolution based auto-encoder performs well because they can learn representations of high-dimensional input image data by extracting salient features and patterns, and efficiently reconstructing the original image from the compressed encoding.
Over-fitting is a problem that occurs when a machine learning model learns too much from the training data and fails to generalize on new or unseen data. This can happen when the model is too complex or the training data is too small. This may lead to poor performance and inaccurate predictions on new data.
Over-fitting can be found by plotting training and validation loss. Increase in validation loss with decrease in training loss can be a sign of over-fitting, as the model does generalize well to the unseen data during training phase. The plot of training and validation loss against number of epochs (Convolution based auto-encoder) is shown in figure 2.5. Based on the plot, it is evident that our model is not over-fitting the training data and it generalizes well for the unseen data.

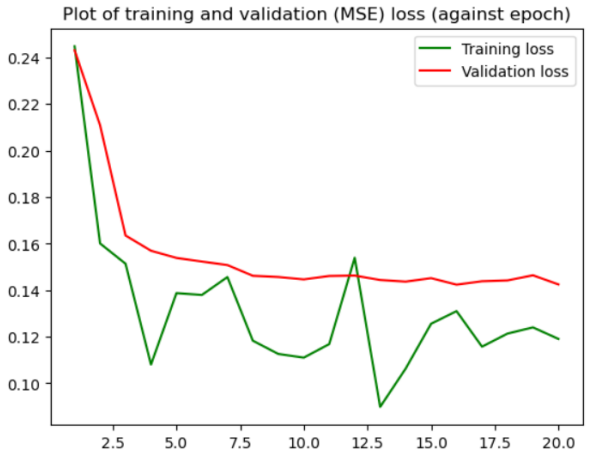

Figure 2.5: Plot of training and validation loss against number of epochs

Question 8 Based on the loss distribution of normal and noisy images, it can be seen that there is a large difference between the MSE loss (mean) values for normal and noisy images. However, this gap may increase or decrease reduce if we vary the amount of Gaussian noise by varying its parameters (mean and variance).
In order to detect anomalies, we can choose a threshold value that is slightly greater than the maximum loss value for normal images. By this way, we can differentiate between normal and corrupted images. The threshold value is given by threshold = max(loss of normal images) + (max(loss of normal images) - mean(loss of normal images)). The threshold value for the latest run, can be visualized in threshold value

graph shown in figure 2.6. (Note: This value may change for each run, but it is automatically calculated)

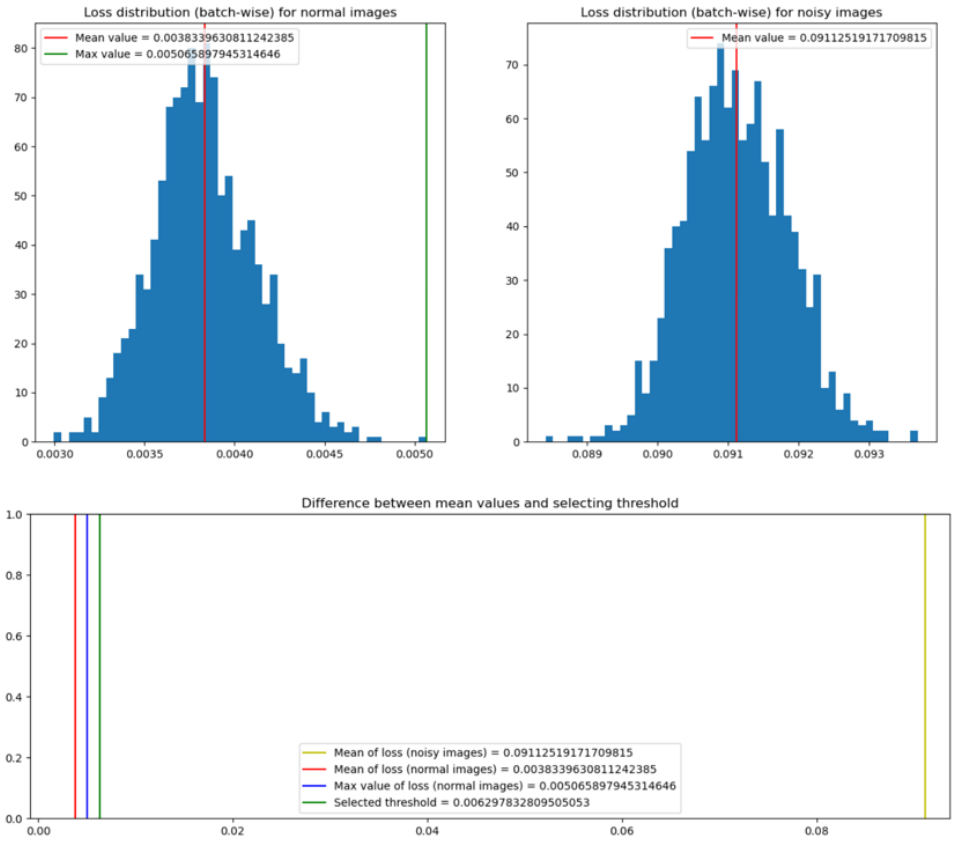

Figure 2.6: Loss (MSE) distribution for normal image (top left), noisy images (top right) and selected threshold value(bottom)

Question 11

ROC (Receiver Operating Characteristic) curve is used to measure the performance of a model by plotting true positive rate (TPR) on y-axis against false positive rate (FPR) on x-axis, at different classification thresholds. The false positivity rate and true positivity rate are given by FPR $= \frac{FP}{FP+TN}$ and TPR $= \frac{TP}{TP+FN}$. The ROC curve for different performances of the classifiers can be seen in figure 2.7

Figure 2.7: ROC curve for different performances of the classifiers

AUC (Area Under the Curve) is a metric that quantifies the area under the ROC curve . The maximum value of AUC is 1. An excellent model has AUC near to the 1 which means it has a good measure of separability whereas a poor model has an AUC near 0.
For our auto-encoder model, we can plot the ROC curve and find AUC values for different threshold value and based on the plot, we can choose a value that provides best performance.

Question 12

FAR - False Alarm Rate (also known as False Positive Rate or fall-out ratio) is the ratio between the number of negative events wrongly categorized as positive (false positives) and the total number of actual negative events (false positives + true negatives).

$$\text{False Alarm Rate (FAR)} = \frac{FP}{FP + TN}$$

where $N = FP + TN$, is the total number of ground truth negatives.
The false alarm rate is an important parameter in anomaly detection. It reflects the trade-off between sensitivity and specificity of the detection method. Sensitivity is the metric that evaluates a model's ability to predict true positives of each available category. Sensitivity $= \frac{TP}{TP+FN}$
Specificity is the metric that evaluates a model's ability to predict true negatives of each available category. Specificity $= \frac{TN}{TN+FP} = 1 - FAR$
A high false alarm rate (high False Positives) means that the anomaly detector is very sensitive and prone to false alarms, which can reduce the credibility and overall efficiency of the system. A low false alarm rate means that the detector is too specific and misses some anomalies, thereby compromising the security and performance of the system.
Application and context of the anomaly detection problem must be taken into account while choosing an ideal value for the false alarm rate. For an example, a low false alarm rate is desirable in cases such as in medical diagnosis or fraud detection, where the cost of missing an anomaly is high and false alarm rate is low. In other cases, a high false alarm rate is acceptable, such as in network intrusion detection or video surveillance, where the cost of missing an anomaly is low and thus the false alarm rate can be high. Therefore, the optimal false alarm rate should be determined by considering the benefits and risks of detecting and missing anomalies, and must also include factors such as available resources and constraints of the system.

Question 13

1. **Computationally expensive** The main drawback of using Auto encoders is they have high computational cost. Especially, if we are using fully connected Neural Networks for encoding and decoding, they required a large number of parameters and operations. The computational cost can be reduced if we use Convolution auto encoders, but even then they require a lot of resources to train.

2. **Inability to capture complex patterns** While auto encoders can perform well on capturing low-level patterns on input data, they struggle to capture features if the input data contains complex patterns. This may lead to poor reconstruction of images and loss of information

3. **Difficulty selecting optimal Hyper-parameters** The selection of hyper-parameters such as learning rate, activation function, number of layers, number of neurons per layer etc. These hyper-parameter affects the performance and quality of auto-encoders and require trial and error to find the optimal value. For example, for the auto-encoder in this experiment, the learning rate was selected based on Mean Squared Error loss on training data after doing several experiments.

Other disadvantages includes over-fitting on training data (performing well on training data but poorly on test data), difficulty handling noisy data and requirement of large amount of training data.

Question 14

1. **DBSCAN** Density-based Spatial Clustering of Applications with Noise is a density-based clustering algorithm that can be extended to detect outliers. It groups data points based on their density and can detect outliers or noise based on the distance criterion.

   DBSCAN works by classifying the data points under three different categories given below.

   - **Core points** are data points located within a within a specified neighborhood of a minimum number of other data points.
   - **Border Points** are data points located within the specified neighborhood of a core point but do not posses neighboring points to be considered as a core point.
   - **Outlier / Noisy points** are data points that are neither core nor border points.

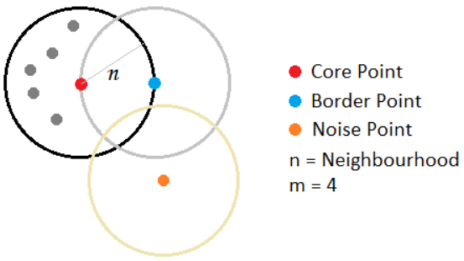

Figure 2.8: DBSCAN clustering for outlier detection

The DBSCAN technique does not require a prior specification of the number of clusters and is ideal for data sets with an unknown number of clusters. It identifies outliers based on their distance of separation from dense data regions.

DBSCAN also has lot of advantages such as automatic cluster detection, robustness to outliers, ability to handle irregular clusters, noise detection, versatility, efficiency, interpretability, scalability, and effectiveness in various data types and dimensions. Hence they are widely used in data-mining, spatial data analysis and machine learning applications.

2. Isolation Forest

Isolation forest is an unsupervised machine learning algorithm and an ensemble method (similar to random forest) which can be used for anomaly detection. It uses the average of the predictions by several decision trees when assigning the final anomaly score to a given data point. Other anomaly detection algorithms define what's "normal" and then report anything else as anomalous, but Isolation Forest attempts to isolate anomalous data points from the get go.

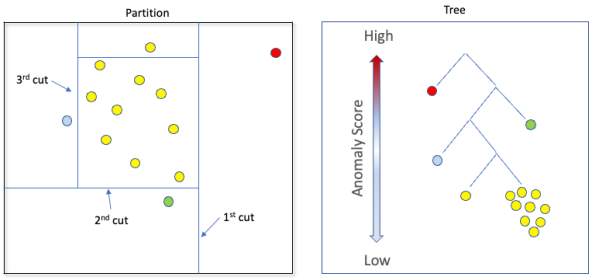

Figure 2.9: Working of Isolation Forest algorithm

Isolation Forests outlier detection are nothing but an ensemble of binary decision trees. A tree in an Isolation Forest is called an Isolation Tree. The algorithm starts with the training of the data and generating Isolation Trees. Below is the algorithm for Isolation Forest

[a] When given a data set, a random sub-sample of the data is selected and assigned to a binary tree.

[b] Branching of the tree starts by selecting a random feature (from the set of all N features) first. And then branching is done on a random threshold (any value in the range of minimum and maximum values of the selected feature).

[c] If the value of a data point is less than the selected threshold, it goes to the left branch else to the right. And thus a node is split into left and right branches.

[d] This process from step 2 is continued recursively till each data point is completely isolated or till max depth(if defined) is reached.

[e] The above steps are repeated to construct random binary trees.

After an ensemble of isolation trees(Isolation Forest) is created, model training is complete. During scoring, a data point is traversed through all the trees which were trained earlier. Now, an 'anomaly score' is assigned to each of the data points based on the depth of the tree required to arrive at

that point. This score is an aggregation of the depth obtained from each of the isolation trees. An anomaly score of -1 is assigned to anomalies and 1 to normal points based on the contamination (percentage of anomalies present in the data) parameter provided.

3 | References

[1] Pranjal Datta. All about Structural Similarity Index (SSIM): Theory + Code in PyTorch — medium.com. https://medium.com/srm-mic/all-about-structural-similarity-index-ssim-theory-code-in-pytorch-6551b455541e. [Accessed 02-01-2024].