**DL-Lab 07**

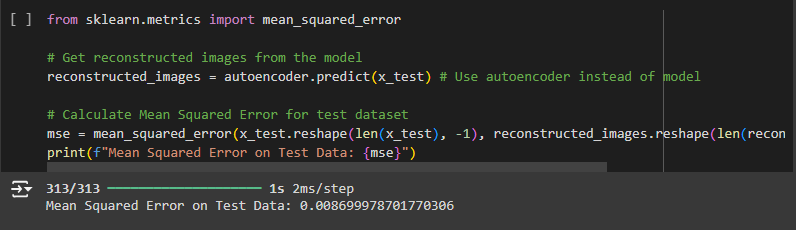
IT21184444

Github repo – <https://github.com/hansika99u/DL-Lab07.git>

Linear Autoencoder vs Principal Component Analysis (PCA)

**I**n a linear autoencoder, the activation functions are removed from both the encoder and the decoder, leaving the network with only linear transformations. In this case, the linear autoencoder becomes functionally equivalent to Principal Component Analysis (PCA).

* A **Linear Autoencoder (AE)** is a type of neural network where no non-linear activation functions (e.g., ReLU, sigmoid) are used in the hidden layers. It functions similarly to **Principal Component Analysis (PCA)** due to the following reasons:
* **Dimensionality Reduction**: Both linear AE and PCA reduce the dimensionality of the data by projecting it onto a lower-dimensional subspace.
* **Linear Transformations**: With no activation functions, the linear AE learns linear transformations like PCA, which projects data onto orthogonal principal components.
* **Reconstruction**: The goal of both linear AE and PCA is to reconstruct the original data from the compressed lower-dimensional representation.
* **Optimization**: While PCA directly computes the principal components via a closed-form solution, linear AE learns this through backpropagation and gradient descent.

1. Mean Square Error Comparison

**A computer screen shot of a program code

Description automatically generated**

A graph of a graph

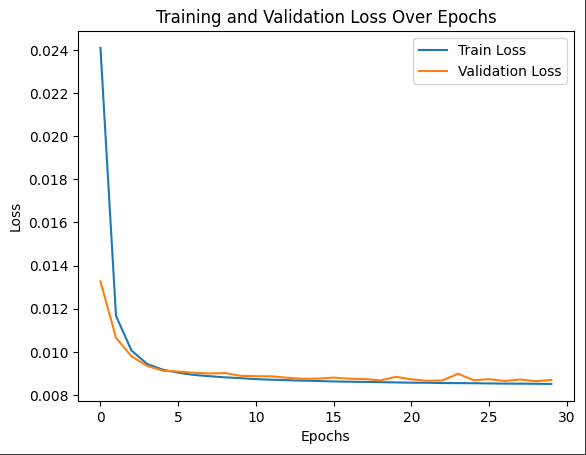
Description automatically generatedGraph Comparison

Figure Cnn Based

Figure Dense based

**Dense-based AE (MSE: 0.0087):** The first model, using fully connected (dense) layers, has a higher Mean Squared Error (MSE) on the test dataset.

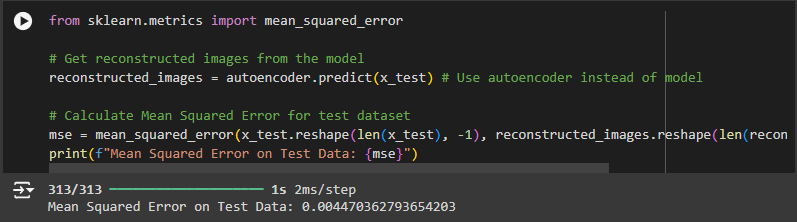
**CNN-based AE (MSE: 0.0017):** The second model, which uses 2D convolutional layers (CNN), achieves a significantly lower MSE. The training and validation loss curves are also smoother, indicating more stable training.

**Conclusion**

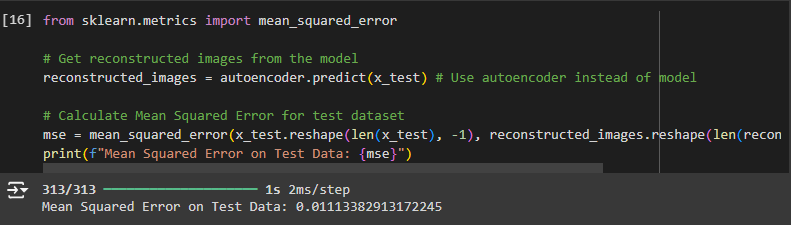
In summary, the CNN-based AE shows better performance over the dense-based AE due to its ability to:

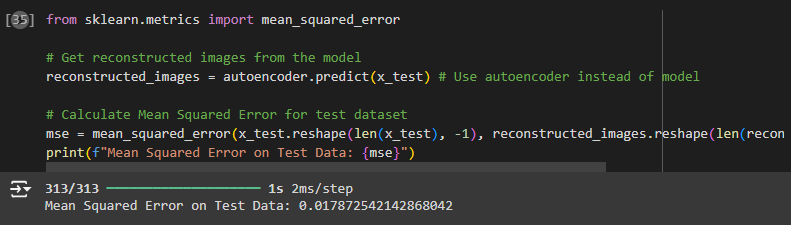
* Capture local features in images more effectively.
* Use parameters more efficiently.
* Learn complex spatial hierarchies through multiple convolutional layers.
* Achieve smoother training with better generalization, reflected in the training/validation loss curves.

A graph of a graph

Description automatically generated“noicy\_factor” when equal to 0.2

A graph of a graph

Description automatically generated“noicy\_factor” equal to 0.4

“noicy\_factor” when equal to 0.6

A graph of a line

Description automatically generated

A graph of a graph

Description automatically generatedA computer screen shot of a program code

Description automatically generated“noicy\_factor” equal to 0.8

After experimenting with different values for the noise\_factor (0.2, 0.4, 0.6, and 0.8), I observed the following pattern:

* The **Mean Squared Error (MSE)** was lowest when the **noise\_factor was set to 0.2**. This indicates that the model was able to reconstruct the images more accurately when the amount of noise added to the images was moderate.
* As the noise\_factor increased (to 0.4, 0.6, and 0.8), the MSE values increased. This suggests that when more noise was added to the images, the autoencoder had a harder time accurately reconstructing the original images, leading to worse performance.

**Best Choice:**

The **best choice** for the noise\_factor is **0.2**, as it resulted in the lowest MSE, indicating better reconstruction performance.

**Explanation:**

* When the noise added to the images is too low (e.g., close to 0), the model might overfit and not generalize well to unseen data.
* When the noise is too high (e.g., 0.8), the model struggles to differentiate between the noise and the true image features, resulting in poor reconstruction quality.
* A moderate amount of noise (in this case, noise\_factor = 0.2) provides enough regularization to prevent overfitting while still allowing the model to learn the essential features of the data for accurate reconstruction.

**Observing Performance Improvements Between the Image De-noising AE and the Vanilla CNN AE**

**Observed Improvements:**

* The **Image De-noising Autoencoder (AE)** shows better performance when reconstructing images from noisy data compared to the **Vanilla CNN AE**. This is evident from the reduced Mean Squared Error (MSE) and smoother loss curves observed in the Image De-noising AE.
* The **Image De-noising AE** is able to generalize better because it has learned to ignore noise and focus on reconstructing the most important features of the original image.

**Reasons for the Improvements:**

1. **Noise as Regularization**:
   * Adding noise to the input during training acts as a form of regularization. This prevents the autoencoder from overfitting to the training data and encourages the model to focus on learning the core features of the data.
   * By learning to denoise images, the **Image De-noising AE** becomes more robust and performs better on unseen data, whereas the Vanilla CNN AE can be more prone to overfitting.
2. **Task Complexity**:
   * The Image De-noising AE has a more complex task compared to the Vanilla CNN AE. It has to reconstruct clean images from noisy inputs, which forces the network to learn more meaningful representations of the data. This leads to improved performance, especially in scenarios with noisy or corrupted data.
3. **Improved Generalization**:
   * Due to its ability to handle noise, the Image De-noising AE generalizes better to new and unseen data. The Vanilla CNN AE, on the other hand, may not generalize as well when the input data is noisy or corrupted since it was trained on clean data.
4. **Regularization by Denoising**:
   * The noise helps the autoencoder avoid learning trivial features of the training data and instead forces the model to learn a compressed, noise-invariant representation. This results in better overall reconstruction performance and smoother loss curves in the Image De-noising AE compared to the Vanilla CNN AE.

Differences Between Autoencoder (AE) and Variational Autoencoder (VAE)

|  |  |  |
| --- | --- | --- |
| **Aspect** | **Autoencoder (AE)** | **Variational Autoencoder (VAE)** |
| **Latent Space** | AE directly learns a compressed representation (latent space) of the input without assuming any distribution. | VAE assumes that the latent space follows a known probability distribution, typically a Gaussian distribution. |
| **Reconstruction** | The decoder reconstructs the original input from the compressed latent space. | The decoder reconstructs the input by sampling from the learned probability distribution in the latent space. |
| **Output** | AE produces deterministic outputs for given inputs. | VAE produces stochastic outputs because it samples from a probability distribution (adding randomness to the reconstruction process). |
| **Loss Function** | AE minimizes reconstruction loss (e.g., Mean Squared Error) directly based on how close the output is to the input. | VAE minimizes a combination of **reconstruction loss** and **KL divergence** (which ensures the latent space follows the assumed distribution). |
| **Learning Representation** | AE may not learn disentangled representations, meaning that the latent variables may not correspond to meaningful aspects of the input data. | VAE tends to learn more meaningful and disentangled representations due to the probabilistic structure in the latent space. |
| **Applications** | AE is typically used for tasks like dimensionality reduction, denoising, and image reconstruction. | VAE is used for generative tasks, such as generating new data, as it can generate new samples by sampling from the latent distribution. |
| **Stochastic vs. Deterministic** | AE has a deterministic encoding process, directly mapping inputs to a compressed representation. | VAE introduces stochasticity, encoding inputs as distributions rather than fixed points in the latent space. |
| **Generative Capabilities** | AE lacks the ability to generate new data samples effectively since it does not model the data distribution. | VAE is a generative model capable of generating new data by sampling from the latent distribution, making it useful for generating new, realistic images. |