Exploring Alternatives to Convolutional Neural Networks for Image Classification

Leo Stepien, Antoine Galas and Yassine Machta
ENSAE Paris

Abstract

This project explores alternatives to Convolutional Neural Networks (CNNs) for image classification tasks. Specifically, we compare CNN-based architectures (VGG, ResNet, UNet) with a Variational Autoencoder (VAE) approach using supervised and unsupervised classification methods. Using MNIST and synthetic datasets, we analyze the models' performance in terms of accuracy, training behavior, and robustness to noise and class imbalance. Furthermore, we investigate the approximation capabilities of these models in the context of optimal Bayes classifiers. You can find the GitHub repository for this project here: GitHub Repository: ML_Model_Comparison

1 Introduction

This project is exploratory and emerged from curiosity about alternatives to Convolutional Neural Networks (CNNs) for image classification. Convolutional architectures excel at extracting features from images. Convolutions, such as Laplacian kernels, can compute the gradient of an image, which is useful for specific tasks. For instance, Frangi et al. Frangi et al. [1998] used gradient and Hessian information to identify tubular shapes , namely vessels, on medical scans.

CNNs extend this concept by learning kernels that extract the features useful for subsequent analysis. When coupled with Multi-Layer Perceptrons (MLPs), CNNs achieve unprecedented accuracy on classification tasks LeCun et al. [1998]. Another significant area of interest is image generation. Although state-of-the-art generation is achieved by models that approach and process images differently (e.g., GANs or diffusion models), earlier models like Autoencoders share similarities with CNNs in how they extract features from images. Some Autoencoder-based architectures even employ convolutional layers Masci et al. [2011] to better capture spatial hierarchies in image data. Among these, Variational Autoencoders (VAEs) Kingma and Welling [2013] remain relevant today, as they learn meaningful latent representations of images, enabling effective image reconstruction. VAEs encode images into a latent space parameterized by a probabilistic distribution, capturing sufficient information for reconstruction tasks.

This project investigates whether the latent space of VAEs could be leveraged for classification tasks. Specifically, we explore supervised classification using an SVM on the latent space and unsupervised classification using K-means clustering.

To benchmark the effectiveness of VAEs for classification, we compare them to three CNN models: VGG, ResNet, and UNet. These architectures represent the evolution of CNNs and encapsulate different eras of state-of-the-art image classification approaches. VGG Simonyan and Zisserman [2015] demonstrates the utility of deeper networks with simple stacked layers. ResNet He et al. [2016] introduced residual connections to address vanishing gradient issues in deeper architectures. UNet Ronneberger et al. [2015], initially designed for biomedical segmentation, showcases the integration of contracting and expanding paths to preserve spatial information. By comparing these models on binary classification tasks using MNIST (0s vs. 1s), we aim to provide insights into the strengths and

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

limitations of VAEs and CNNs. Additionally, we test model robustness on synthetic datasets with controllable distributions of images and labels.

2 Methods

2.1 Models

VGG. VGG Simonyan and Zisserman [2015] is a deep convolutional neural network designed to extract hierarchical features from images by leveraging the principles of convolution. Convolution is a mathematical operation applied to input data (such as an image) using a filter (or kernel). This filter slides over the input, performing element-wise multiplication and summation to produce an output feature map. Convolutions allow the network to detect spatial patterns such as edges, textures, and complex structures by learning different filters in successive layers. Additionally, pooling layers (typically max-pooling) are often employed to downsample feature maps, reducing spatial dimensions and computational cost while retaining critical information.

VGG represents a significant evolution in convolutional architectures by standardizing the use of small 3×3 filters throughout the network. Specifically, VGG-19 consists of 19 layers, including 16 convolutional layers interspersed with max-pooling layers and 3 fully connected layers at the end. The architecture progresses from extracting low-level features in the initial layers to capturing high-level abstract features in the deeper layers. All convolutional layers use Rectified Linear Unit (ReLU) activation, which introduces non-linearity, enabling the network to learn complex representations. The fully connected layers at the end act as a classifier, mapping the high-dimensional features to output predictions. The design simplicity and effectiveness of VGG have made it a foundational architecture for computer vision tasks, serving as a benchmark for many subsequent innovations.

ResNet. He et al. [2016] introduced ResNet, which employs residual learning to address the challenges associated with training very deep neural networks. Prior to ResNet, the primary approach to improving performance in CNNs was to increase their depth by adding more layers. While deeper networks have a greater capacity to learn complex representations, they also face significant challenges, particularly the vanishing gradient problem. As networks become deeper, gradients can become extremely small during backpropagation, making it difficult for the earlier layers to update their weights effectively. Additionally, very deep networks suffer from the degradation problem, where increasing the number of layers can lead to higher training error, as the network struggles to learn an identity mapping when additional layers are unnecessary.

ResNet addresses these issues by introducing residual connections, or "skip connections," which allow the network to learn residual functions $F(x) = H(x) - x$ instead of directly learning the mapping $H(x)$. By reformulating the problem in this way, ResNet makes it easier for the network to optimize very deep architectures. If an identity mapping is optimal for a subset of layers, the residual connection allows the network to simply push the residuals $F(x)$ to zero, bypassing the need to fit an identity mapping with nonlinear layers. This mechanism enables gradients to propagate more effectively, mitigating the vanishing gradient problem and improving convergence. For this project, we built our ResNet by incorporating residual connections into VGG-style convolutional blocks.

UNet. UNet, introduced by Ronneberger et al. [2015], is a convolutional network architecture originally designed for biomedical image segmentation. The architecture is structured in two main parts: a contracting path (encoder) and an expanding path (decoder). The contracting path extracts hierarchical features through successive convolutional and pooling layers. In the early convolutional layers, the network captures low-context features, such as edges, shapes, and basic textures. As the network goes deeper, the later layers focus on extracting high-context features, which involve more complex patterns and semantic information, such as identifying specific objects or regions in the image. The decoder mirrors the encoder in structure, using up-convolutional (transpose convolutional) layers to progressively upsample the feature maps and reconstruct spatial information.

In traditional CNNs, these low-context features from the upper layers are effectively discarded as the network focuses on the high-context features for classification or other tasks. UNet addresses this limitation through its skip connections, which link corresponding layers in the contracting and expanding paths. These connections transfer the low-context features extracted in the early stages of the encoder directly to the decoder, allowing the network to preserve and utilize fine-grained

spatial details, making UNet particularly effective for tasks where precise localization and detailed reconstruction are critical.

For this project, we adapted UNet to use a ResNet-based encoder to compare the two more easily.

Variational Autoencoder (VAE). Variational Autoencoders (VAEs) are a class of generative models that learn to encode data into a probabilistic latent space, enabling both image reconstruction and data generation. In a VAE, the encoder models the posterior distribution $q(z|x)$, where x is the observed image and z is the latent variable. This posterior distribution is approximated as a Gaussian with mean $\mu(x)$ and standard deviation $\sigma(x)$, which are produced by the encoder network. The encoder network outputs the parameters $\mu(x)$ and $\sigma(x)$, which define the distribution over the latent space for a given input image x.

The decoder, in turn, models the likelihood of the image given the latent variable, represented as $p(x|z)$, which is typically modeled as a Gaussian distribution with a fixed variance. The decoder reconstructs the image by sampling from this latent space distribution. To optimize the VAE, we minimize the variational lower bound, which consists of two parts: the reconstruction loss (often pixel-wise binary cross-entropy or mean squared error) and the Kullback-Leibler (KL) divergence between the learned posterior $q(z|x)$ and the prior $p(z)$ over the latent variables, which is typically chosen to be a standard normal distribution $\mathcal{N}(0, I)$.

We experimented with a VAE using Multi-Layer Perceptrons (MLPs) for both the encoder and decoder, as well as versions incorporating UNet-style encoders and decoders. For classification tasks, we utilized the latent space in two ways: supervised classification by adding a linear layer to the latent space followed by an SVM classifier, and unsupervised classification by applying K-means clustering to the latent representations.

2.2 Experimental Design

MNIST Experiments. In this work, all models were trained for binary classification on the MNIST dataset, specifically to classify digits 0 and 1. We employed several evaluation metrics, including training loss, training and validation accuracy, training time, and gradient norms, to track the model's performance and stability. For the CNNs, we used Binary Cross-Entropy (BCE) loss to optimize the classification task. On the other hand, the VAE followed a two-step process: initially, we trained the VAE on its reconstruction task using KL divergence and Mean Square Error (MSE). For classification, if the VAE was coupled with a SVM classifier it needed training, we used the hinge loss given by $L(y) = max(0, 1 - ly)$ where l is the label of the image and y the output logits of the linear layer.

To assess the models' robustness, we introduced two challenges: noise and class imbalance. We simulated noisy inputs by adding white noise and cropping the images randomly and tested the model's resilience to such distortions. For class imbalance, we manipulated the training data to skew the distribution of class 1 (digit "1") relative to class 0 (digit "0"), in our case 90 % of the images were of class 1, and evaluated how well the models performed under these conditions. This allowed us to explore how each model handles these common real-world challenges.

For the VAE, we further evaluated the quality of the generated images by comparing their distribution to the known distribution using the sliced Wasserstein distance which is often for penalizing generative models. The 1 dimensional Wasserstein distance between two distributions μ and ν is given by

$$W_1(\mu, \nu) = \int_0^1 |F_\mu^{-1}(u) - F_\nu^{-1}(u)|du \tag{1}$$

where F_p^{-1} is the inverse cumulative distribution function of the distribution p. We obtain the sliced Wasserstein by projecting the distributions on the unit sphere the same dimension as the samples from the distribution. Formally the sliced Wasserstein is written,

$$SW_1(\mu, \nu) = \int d\sigma_S(v) W_1(v\#\mu, v\#\nu) \tag{2}$$

where $v\#\mu$ is the distribution of $v^T.X$ where X is distributed according to μ and σ_S is the normal distribution over the unit sphere. We could then measure the distance between the actual images of digits and the ones created by the VAE.

Synthetic Dataset Experiments. Synthetic datasets were constructed using Gaussian distributions with distinct parameters for each class. This setup enabled us to compute the theoretical Bayes classifier for the data, which served as an optimal decision rule. By approximating the risks of different models by their empirical counterparts, we were able to study the trade-off between estimation error and approximation error in the excess risk decomposition. Recall the excess risk decomposition:

$$R(\hat{h}) - R(h_*) = R(\hat{h}) - R(\bar{h}) + R(\bar{h}) - R(h_*) \tag{3}$$

where we follow the notations of the lecture.
We approximate all risks by the empirical risk on a set different from the one $\hat{h}$ is computed from, we then have:

$$EE = R_n(\hat{h}) - R_n(\bar{h}), \tag{4}$$

$$AE = R_n(\bar{h}) - R_n(h_*) \tag{5}$$

$\bar{h}$ is obtained by training the model on the set used for the empirical approximation, this assumes:

$$\arg\min_{h\in\mathcal{H}} R_n(h) \subset \arg\min_{h\in\mathcal{H}} R(h) \tag{6}$$

which is generally not true, it is rather the opposite inclusion that holds.
As for h_*, given that we know the distribution of the images, we can compute it as follow:

$$h_*(x) \in \arg\max_{y\in\{0,1\}} \mathbb{P}(Y = y|X = x) \tag{7}$$

with

$$\mathbb{P}(Y = y|X = x) = \frac{\mathbb{P}(X = X|Y = y)\mathbb{P}(Y = y)}{\mathbb{P}(X = x)} \tag{8}$$

Only the top term depends on Y and interest us, by construction in our experiment we have $\mathbb{P}(Y = 1) = \mathbb{P}(Y = 1) = 0.5$ and

$$P(X = x \mid Y = c) = \frac{1}{(2\pi)^{d/2}|\Sigma_c|^{1/2}} \exp\left(-\frac{1}{2}(x - \mu_c)^\top \Sigma_c^{-1}(x - \mu_c)\right) \tag{9}$$

Given the decision boundary $\mathbb{P}(X = x|Y = 0) > \mathbb{P}(X = x|Y = 1)$ we take the logarithm to simplify:

$$-\frac{1}{2}(x - \mu_0)^\top \Sigma_0^{-1}(x - \mu_0) - \frac{1}{2}\log|\Sigma_0| \;>\; -\frac{1}{2}(x - \mu_1)^\top \Sigma_1^{-1}(x - \mu_1) - \frac{1}{2}\log|\Sigma_1| \tag{10}$$

Canceling shared terms like $-\frac{1}{2}\log(2\pi)^d$, the decision boundary simplifies to:

$$(x - \mu_0)^\top \Sigma_0^{-1}(x - \mu_0) - (x - \mu_1)^\top \Sigma_1^{-1}(x - \mu_1) \;<\; \log\left(\frac{|\Sigma_1|}{|\Sigma_0|}\right) \tag{11}$$

This is the formula we use in the implementation.

3 Results

3.1 Training on MNIST

The accuracy achieved by each model when training on the MNIST images is reported in Table 1, we see that VAE with classification done on the latent space yield similar results as the CNNs when the images are left untouched and when we integrate class imbalance. Although in a noisy setting the VAE we used was unable to extract any meaningful information from the noisy images, the model

Table 1: Validation Accuracy (%)

Model	No Noise (%)	Class imbalance (%)	Noisy Images (%)
VGG	99.91	96.50	93.62
Resnet	99.95	**99.90**	96.88
Unet	**100.00**	**99.90**	96.97
Unet Pretrained	99.95	**99.90**	**97.26**
VAE + SVM	98.16	97.31	51.06 (65.90)
VAE + KMeans	99.05	99.38	50.49 (61.37)

Table 2: Training Time (seconds)

Model	No Noise	Class imbalance	Noisy Images
VGG	1000.81	1026.58	1336.70
Resnet	1060.90	1095.60	1416.24
Unet	1620.32	1642.55	1961.61
Unet Pretrained	1459.54	1460.88	1793.42
VAE + SVM	381.66	380.04	1265.36
VAE + KMeans	**256.94**	**270.36**	**707.78**

trained on untouched images to make predictions on noisy images yielded slightly better results (see the values in parenthesis in the right column). This results shows the weakness of the MLP encoder of our VAE which is not optimal for image analysis, unfortunately our experiments trying to inherit the UNet encoder and decoder for the VAE were unsuccessful.

We see in Table 2 that the VAE model we used is much smaller than the CNNs as the training times are much less, especially in the fully unsupervised setting when we use K-Means clustering on top of the latent space. It is also interesting to see the increasing complexity of the CNNs as the UNet trains slower than the ResNet that train slower than the VGG. Mainly due to adding parameters but it also translates into better results as we can see in Table 1 in the right column.
We recorded the loss, gradient norm and accuracy of our models through the epochs of training. Figure 1 shows these metrics when training all CNNs and the SVM classification layer of the VAE on noisy images. We can see the VAE failing to classify and also see that each generation of CNN converges faster that the previous one which counterbalances the fact that bigger models take more time to train by epoch. We also see that the validation accuracy stays close to the training accuracy which reassures us into thinking that our models do not overfit.

By looking at the loss and its component during training of the VAE Figure 2 it is clear why it failed with noise Figure 3. The reconstruction loss is off the chart because the noise makes it impossible to reconstruct the same image unless the model learns to model the noise which would be overfitting.

3.2 Visualizing learned information

To have a better understanding of what are the features extracted by the convolutions of CNNs we can look at the kernel that extract features and at the features maps. Kernels shown in Figure 4 are the ones used in the first convolution of our VGG after training, they are what is learned by the model. These kernels are difficult to interpret so we look at the feature maps which correspond to the result of applying these kernels to the image and we obtain Figure 5. It can be seen on these features maps that what is being extracted by the first layer are contours in the image whereas the features extracted by the last layer are much more abstract (see Figure 6)

For our VAE implementation where the encoder and decoder are MLPs it is impossible to have a visual representation of the features extracted by the image (we tried reusing the encoder and decoder of UNet to build our VAE but did not manage to make it work). Yet, we can look at the latent space and especially compare the representations of zeros and ones to show why the approach of separating the latent space with a SVM or clustering works. After computing the latent representation of somes images of zeros and ones they were projected on a 2D space using t-SNE projection resulting in Figure 7.

Figure 1: Training behaviors of the models when trained with noisy images.

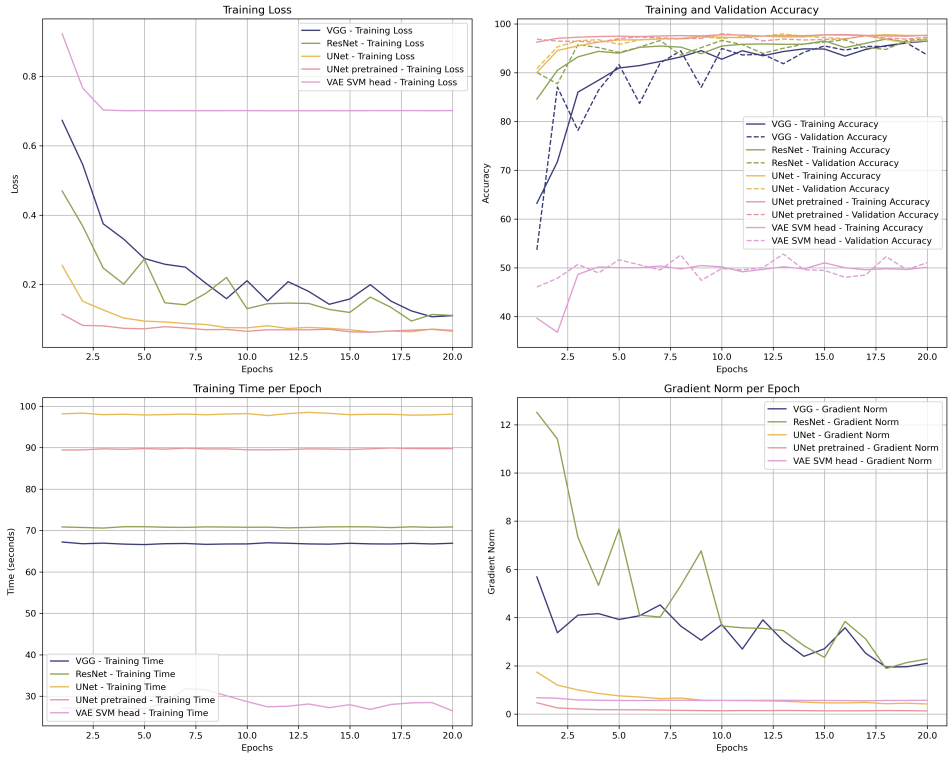


Figure 2: VAE loss without noise.

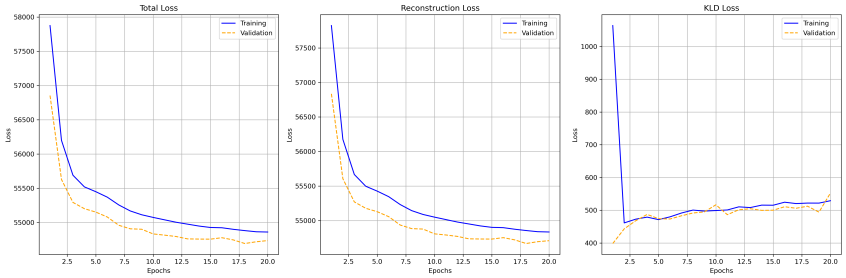
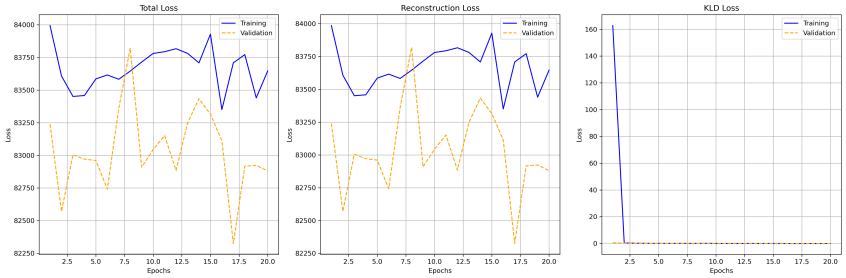


Figure 3: VAE loss with noise.



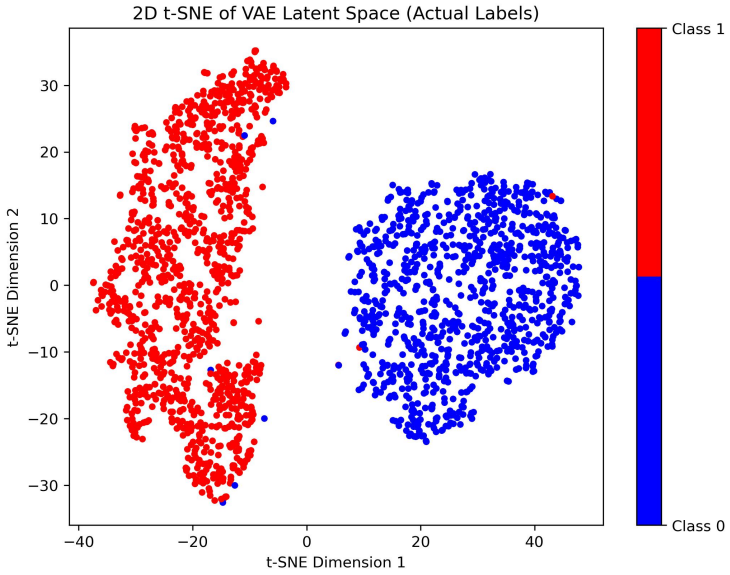
We can see a clear separation between zeros ans ones and it is what is leveraged by the SVM to do classification and why clustering has good results. On the other hand we refer you to Figure 17 in the appendix that shows the same plot when the VAE is trained with noisy images to see why the classification was impossible as the clusters are merged. Figure 8 are plots of each approach, on the left the labels obtained through KMeans clustering on the higher dimensional space are showed in

Figure 4: Some kernels from VGG first layer.

Figure 5: Feature maps extracted on images of zeros by the first layer of the VGG.

Figure 6: Feature maps extracted on images of zeros by the last layer of the VGG.

Figure 7: VAE latent space projected in 2D with groundtruth label of each point.



the 2D projection, on the right, points that are in the hyperplane $< w|x >= b$ where w and b are the SVM parameters, where projected along with the latent representations.

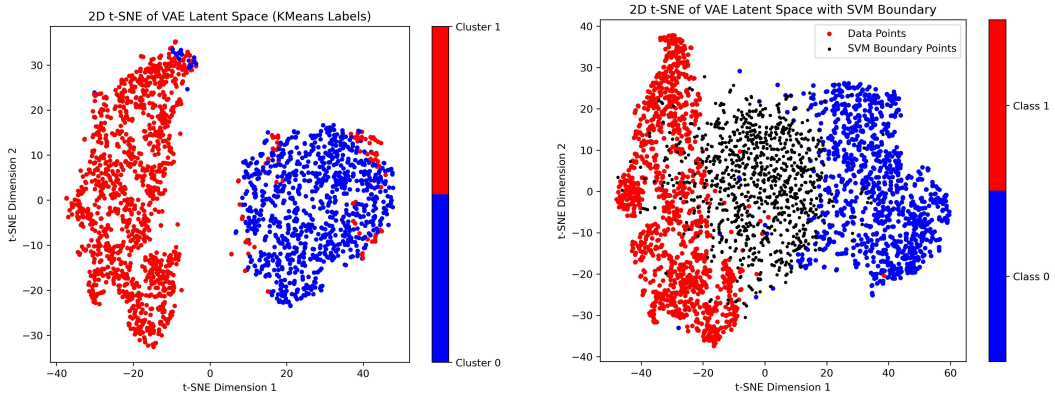
We can see that the cluster labels align with the actual labels of Figure 7 and that the points from the SVM hyperplane fall between the two clusters of representations of ones and zeros.
A testimony to how well the VAE learns to extract features from an image is the quality of the reconstructed images, as it shows on Figure 9 it seems the model is doing a good job.

To have a measure of how well the model reconstruct the images we computed the Wassertein distance between the original images of the test set and their reconstruction. In the context of VAE it measures the distance between $p(x)$ and $p(x|z)$. For zeros the distance was **0.065** and for ones **0.0395** as a reference the distance between the original images of zeros and ones was **0.152**, so it corroborates what can be seen qualitatively on the plot.

3.3 Error analysis

Now we dive into the analysis of the excess risk decomposition, we trained our models to differentiate between an gaussian multivariate with same variance $3I_{32\times32}$ and mean -1 or 1. As explained in

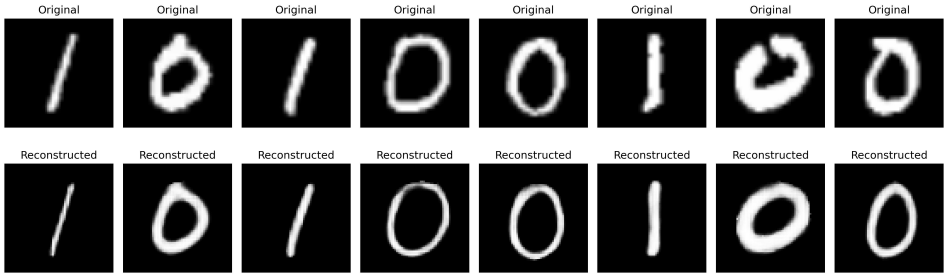
Figure 8: Visualization of each classification method on VAE latent space.



(a) Labels obtained with KMeans.

(b) SVM boundary points projection in latent space.

Figure 9: Original images on top, reconstructed images by the VAE below.



Section 2.2 knowing the distribution of the images we can compute the bayes estimator. Terms in the excess risk decomposition formula are approximated by their empirical counterpart by computing on a large test set. The plot of the estimation error, approximation error trade off is often given with the cardinal of the set of possible classifiers as x-axis instead we replicated the errors against the number of parameters of the models. This assumption is corroborated by a result proven in Bartlett et al. [2021] on the lower bound of the VC dimension of neural networks that use ReLu activations between layers which is our case. The result states, with W the number of parameters and L the number of layers:

$$VC \geq \frac{WL}{K} \log \frac{W}{L} \tag{12}$$

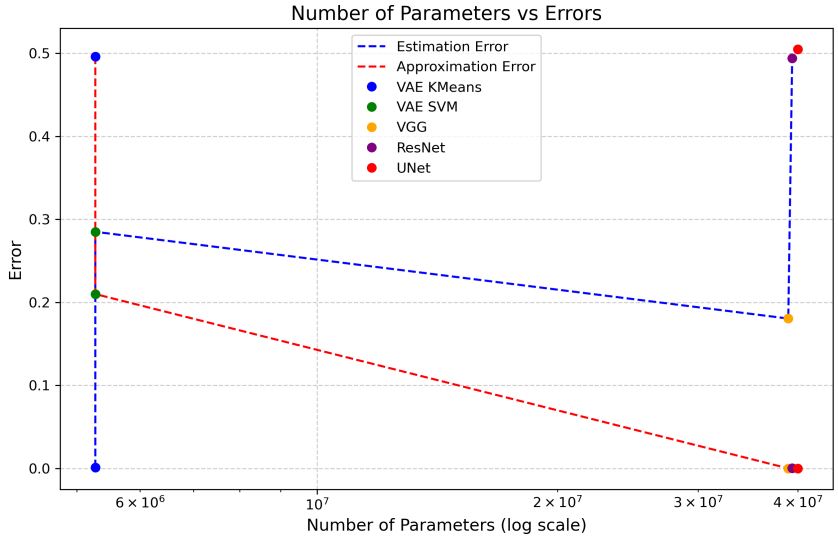
Where $W > KL > K^2$, differentiating with respect to the number of parameters gives:

$$\frac{L}{K}\left(\log \frac{W}{L} + 1\right) \geq 0 \tag{13}$$

Hence the model complexity increasing with the number of parameters, so having the number of parameters as x-axis captures the same idea as the size of possible classifiers.

Figure 10 illustrates the phenomena that the more complex the model, the closer it can get to the optimal bayes classifier resulting in smaller approximation error (red line). Conversely the more complex the more prone to overfitting is the model resulting in greater estimation error (blue line), this also corroborates the main upper bound of VC theory as the the bound on the estimation error grows according to $\sqrt{VC \log \frac{1}{VC}}$. To obtain these results we pushed the trait of overfitting by giving only a handful data points for training. We can see that the VAE + SVM offers a better trade-off of errors with fewer data points.

Figure 10: Estimation errors and Approximation errors of our models.



4 Conclusion

This project demonstrated that latent spaces from VAEs can serve as effective representations for classification tasks. Although they do not outperform CNNs in accuracy and in our case our implementation could not deal with noisy images, it was easier to train and could be used without any labels if paired with K-Means. The VAE approach has to be tested on more difficult tasks and we think the MLP implementation would soon be limited, switching to convolutional encoders and decoders would probably result in better performance and robustness to noise but the model would be slower to train and prone to overfitting has the CNNs were. Yet, we have proven that the latent space is very valuable, regardless of the surrounding architecture, and could be leveraged to have an unsupervised classification model.

References

Peter L Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. *arXiv preprint arXiv:2107.01774*, 2021. URL `https://arxiv.org/abs/2107.01774`.

Alejandro F Frangi, Wiro J Niessen, Koen L Vincken, and Max A Viergever. Multiscale vessel enhancement filtering. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 130–137. Springer, 1998.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International Conference on Artificial Neural Networks*, pages 52–59. Springer, 2011.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

5 Appendix

All these figures can be obtained from our repository GitHub Repository: ML_Model_Comparison. Please refer to subsection 5.1 for instructions

5.1 GitHub Usage

Start by setting up the environment by running:

```
pip install -r requirements.txt
```

Setup the module ClassComp by using:

```
pip install -e .
```

Use the notebooks in order:

- **I- Train_Gaussian_noise**: Runs reconstruction experiments on synthetic Gaussian Dataset and extracts results and metrics $\sim$ 2 hours (We run 10 times on 10 training images and 1000 validation ones to better estimate R_n).
- **II - Train Mnist**: Runs reconstruction experiment on Mnist dataset (For 10 epochs $\sim$ 3 hours per dataset modification for all models i.e., Vanilla, noisy+class imbalance, class_imbalance ratio $= 0.9$) for 20 epochs, it doubles.
- **III - Plot_graphs**: Plots results from Mnist-based models and metrics.

In case you wish to check the plotting without fully training, we have also committed and pushed a bunch of weights and results you can check out in **III - plot_graphs**.

5.2 Training additional figures

Figure 11: Training behaviors of the models when trained with normal images.

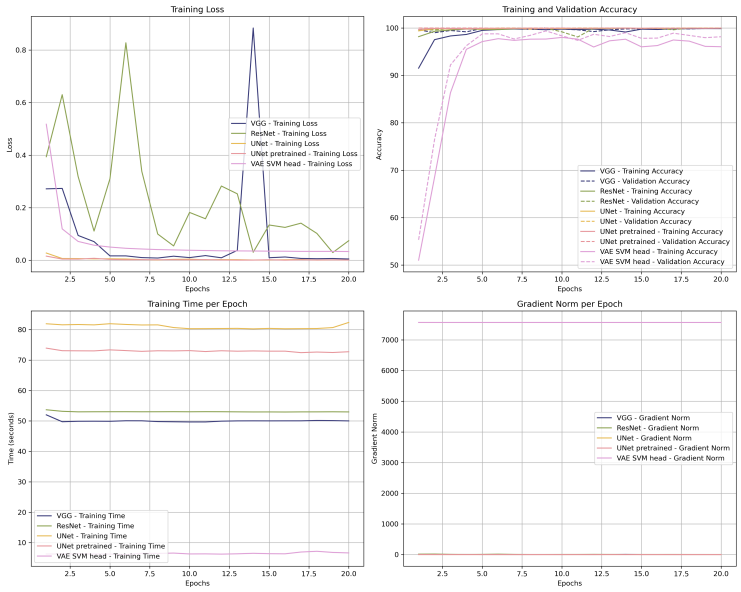


Figure 12: Training behaviors of the CNNs when trained with noisy images.

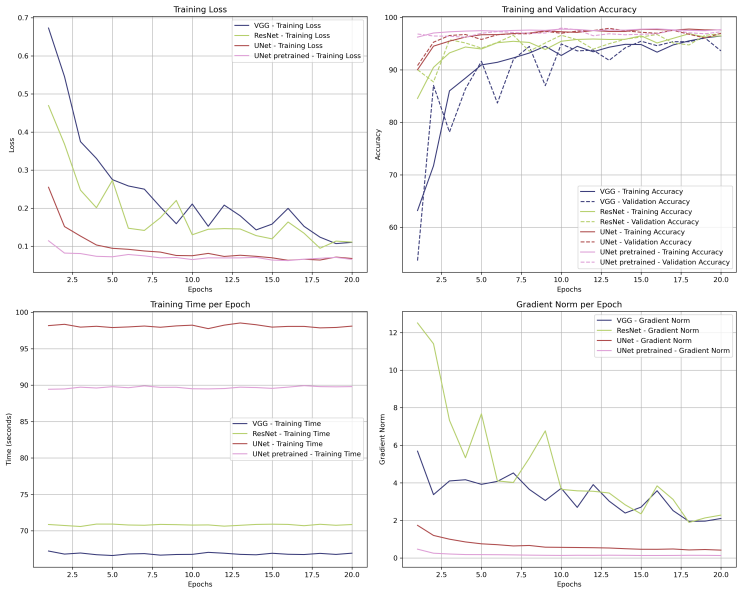
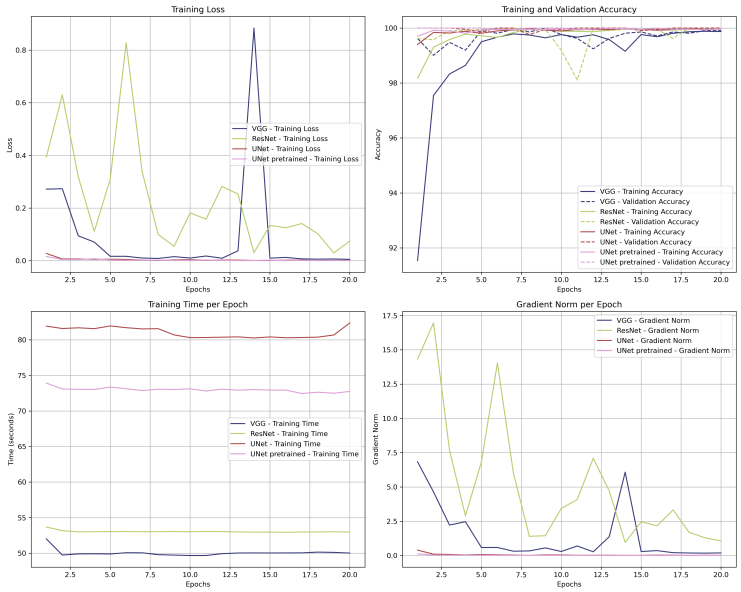


Figure 13: Training behaviors of the CNNs when trained with normal images.



5.3 Visualization additional figures

Figure 14: Feature maps extracted on images of ones by the first layer of the VGG.

Figure 15: Feature maps extracted on noisy images of zeros by the first layer of the VGG.

Figure 16: Feature maps extracted on images of zeros by the first residual layer of the ResNet.

Figure 17: VAE latent space projected in 2D with groundtruth label of each point with noisy images.