Study of Generative Models

Yug Ajmera, Divyanshu Sahu, Kausik Sivakumar, Vanshil Shah

***Abstract*—The project aims to compare different generative models, that utilizes encoder-decoder architectures. In this study, we fit three different models - Principal component analysis, Autoencoders and Variational autoencoders- on CIFAR 10 dataset. We perturb their latent space in three ways, and compare the image generated. During the course of the study, we found that autoencoders were able to generate a few synthetic images that were visually believable. The study concludes by comparing these models by passing the synthetic image generated through a pretrained Alexnet built for CIFAR 10 classification, and proves that the autoencoders are better in generating better synthetic images than that of PCA and the baseline**

I. Motivation

We have seen that unsupervised methods help project p dimensional image data to d dimensions (where d could be lesser than p). In this latent-space representation, images belonging to a particular class are closely clustered. We have seen algorithms like PCA which capture linear feature relationships and autoencoders which capture non-linear feature relationships. Furthermore, algorithms such as variational autoencoders also learn the statistics of the class of the image. Our hope is to reconstruct an image belonging to a class from an unseen datapoint in latent-space by adding some noise in these loadings belonging to a particular cluster, thus attempting to generate synthetic images. We propose on doing a comparative study of different model that could encode image data in a few dimensions, perturb and reconstruct them from the latent space loadings. The scope of this study would be limited to generating synthetic images from PCA, autoencoders and variational-autoencoders. A baseline metric for this could be an average image of the data belonging to all classes

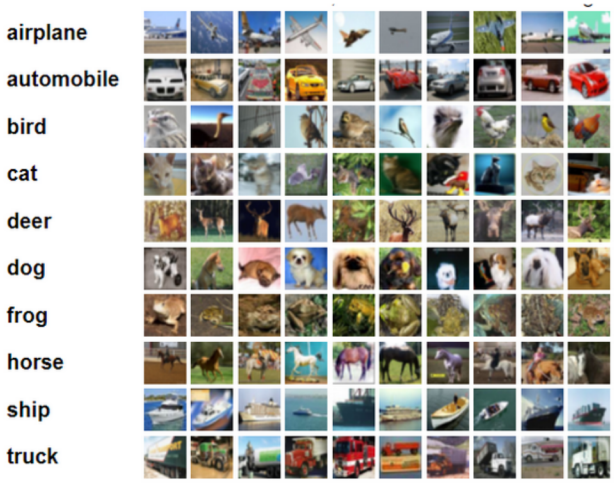
II. Related Work

The relevant work to our project outlines are the studies and methods, which perform dimensionality reduction in feature space and interpolate the latent vectors in the latent space. These algorithms also mention decoding of the latent space to feature space while maintaining as much information as required

1) This blogpost studies in brief the way of using PCA for latent space decomposition - **PCA for LSD**.
   In the above article, the author has performed a PCA on MNIST dataset. We could try infering different stratergies and we hope this would serve as a good starting point for building upon this method to attempt image generation
2) Further, we would try to explore methods which learn a non-linear mapping like deep generative models. In this article the author has compared the pros and cons of autoencoder and VAE - **VAE**
   After testing our approach using Auto Encoder, we would try to go along the lines of what the author has proposed. By using VAE, we will try to get a continuous latent space, in which the interpolation will be easier and more predictible than that of an AE.

III. Dataset

This study utilizes the CIFAR-10 dataset, which consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images - **CIFAR 10**



IV. Problem Formulation

The analysis that we are trying to do as part of our project is to study the effect of using different unsupervised models to generate data by reconstructing the perturbed latent space loadings. The machine learning task would be to -

1) Build a model that best encodes the data to a few dimensions and make a bottleneck model (while capturing as much as variance possible between features of the input datapoints)
2) Perturbing the latent space in three ways
   - Perturb the latent space scores of the training data with by randomly sampling from a gaussian with zero mean and a small variance, where the perturbed image would be the "synthetic data" generated
   - Perturb the latent space scores stochastically towards that belonging to the class of another image
   - Perturb the latent space scores towards the approximate average of the latent space loadings of the class

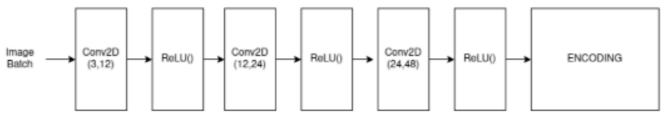
3) Reconstruct images after perturbation and evaluate the beliefness of this reconstructed synthetic image by comparing softmax outputs from a pretrained network [Yang(2019)]

Model	Params (M)	CIFAR-10 (%)	CIFAR-100 (%)
alexnet	2.47	22.78	56.13

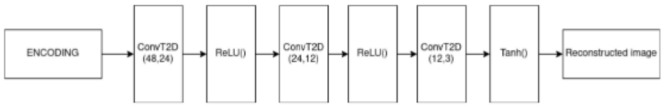
4) This would be done for different models mentioned - PCA, autoencoders and variational autoencoders to finally score based on softmax outputs of the pretrained network's logits
5) We try to compare generative models, and hence we focus on those that could capture as much information in some manifold embedding, so that reconstruction and inference is possible. Thus, we went with these models which have been proven to capture as much variance possible within the features in the latent space

V. Methods

1) Baseline: Our baseline is the average image of each class. We believe this to be a good baseline because, it tells us how an average car or a horse looks whereas our model (PCA, autoencoders) tries to capture rich features than just the average in pixel space.
2) PCA
   - PCA is the next best method to consider after an average for reconstruction, because it tends to cacpture the variance across my image features in orthonormal components
   - It captures linear relationship of features to the latent embedding which best explains my images
   - We are using sklearn's implementation of PCA [Scikit-PCA(2013)]
3) AE
   - After evaluating linear mapping, we find non-linear relationship in features by using autoencoders [Bank et al.(2021)Bank, Koenigstein, and Giryes] [Chenjie(2019)]
   - We use PyTorch to implement the network. This contains an encoder to encode images to its latent embeddings, and a decoder to reconstruct images back from the latent spacce
   - Encoder: The encoder consists of three convolution layers followed by a ReLU layer, except for the last convolution layer which is followed by a Tanh() function



   - Decoder: The decoder consists of three transpose convolution layers followed by ReLU()



Algorithm 1 AUTOENCODERS

model = AutoEncoder()
Inputs: img
Outputs= reconstruction
Loss = MSE(img,reconstruction)
Optimizer = Adam
model.params() = weights
Objective = $\min_{weights}(Loss)$
for $i \leftarrow 1$ *to epochs* **do**
  for *imgs in batch* **do**
    reconstructions = model(imgs)
    loss = MSE(imgs,reconstructions)
    weights $\leftarrow \eta * -\nabla_w loss$

4) VAE
   - The next upgrade from an autoencoder is a variational autoencoder. This generative model tends to not only decrease the reconstruction error but also tries to learn the statistics of the image's distribution in the latent space by approximating it as a gaussian. Thus there is a KL divergence loss included to try fit the latent space distribution close to that probability distribution in latent space to generate the dataset [Kingma and Welling(2014)]
   - We standardize the image to mean 0 and variance 1, and try to fit the binary cross entropy loss to take care of the reconstruction error (looking at the problem as p categorical features that take value of 0 or 1) [Malysheva(2018)]

```
[6]  1 model

vae(
  (encoder): Sequential(
    (0): Conv2d(3, 32, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): Conv2d(32, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (4): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU()
    (6): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (7): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ReLU()
  )
  (q_mean): Linear(in_features=2048, out_features=128, bias=True)
  (q_logvar): Linear(in_features=2048, out_features=128, bias=True)
  (project): Linear(in_features=128, out_features=2048, bias=True)
  (decoder): Sequential(
    (0): ConvTranspose2d(128, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU()
    (3): ConvTranspose2d(64, 32, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (4): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): ReLU()
    (6): ConvTranspose2d(32, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1))
    (7): BatchNorm2d(3, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ReLU()
    (9): Sigmoid()
  )
)
```

Algorithm 2 VAE

model = VAE()
Inputs: img
Outputs= reconstruction
BCE= -$\sum_i x_i log(\hat{y_i}) + (1 - x_i) log(1 - \hat{y_i})$
ELBO =$-0.5 \times \sum_i (1 + log(var) - mean^2 - var^2)$
Loss = $BCE + KL$
Optimizer = $Adam$
model.params() = $weights$
Objective = $min_{weights}(Loss)$
for $i \leftarrow 1$ *to* $epochs$ **do**
  for $imgs$ *in* $batch$ **do**
    reconstructions = model(imgs)
    weights $\leftarrow \eta * -\nabla_w loss$

5) Perturbations
Perturbations are generated three ways as specified in the problem formulation

- Random perturbation: For the given set of latent space variables, we add a Gaussian noise to these latent scores and reconstruct the images back using the respective method.

$$z_{random} = z + \mathcal{N}(\mu, \sigma^2)$$

- Stochastic Score perturbation: For a given set of latent space variables, we take two latent space scores for those two variables and then do a linear interpolation from one score to the other. We use the updated score to decode and reconstruct the image back using the respective method.

$$z_{stochastic} = z_{class_a} + (1 - \alpha) z_{class_b}$$

- Towards class average (for class i)

$$z_{class_avg} = z_{class_i} + (1 - \alpha) z_{centre\_class_i}$$

6) Reconstruction

$$reconstructed\_img = decoder(z)$$

VI. Experiments and Results

A. Results

1) Baseline
The average images are blurry and show that a simple average image of a particular class would not able to represent the images for any reconstruction and is unable to capture any low dimensional features.

Fig. 1. Original image and reconstructed image

2) PCA
a) The results of the reconstructed images after adding stochastic gaussian noise to the generated scores is as follows:

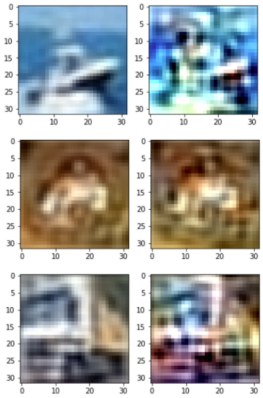


Fig. 2. Original image and reconstructed image

b) The results of the reconstructed images after doing linear interpolation between any two random classes:

Fig. 3. Image of first class, Image of second class and reconstructed image of the interpolated score

c) The results of the reconstructed images after moving towards the cluster centre:

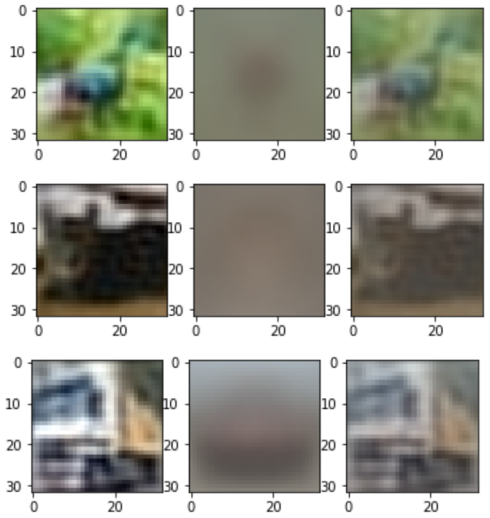


Fig. 4. Image of a class, Image of cluster center belonging to that class and reconstructed image

3) AE

a) The results of the reconstructed images after adding stochastic gaussian noise to the generated scores is as follows:

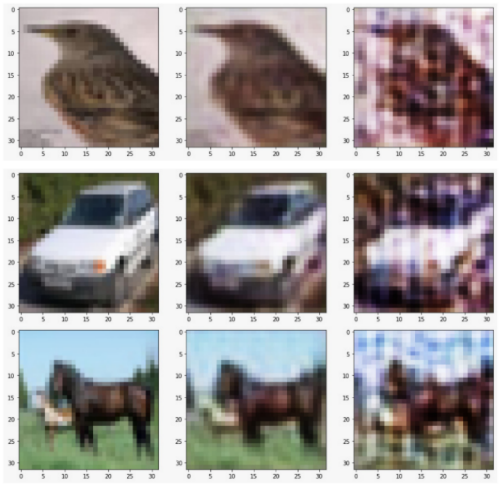


Fig. 5. Original image and reconstructed image

b) The results of the reconstructed images after doing linear interpolation between any two random classes:

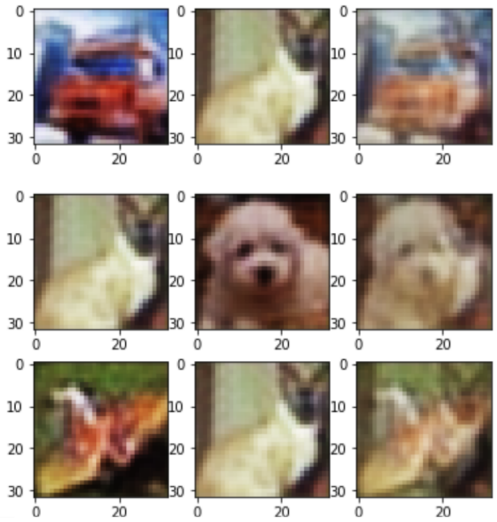


Fig. 6. Image of first class, Image of second class and reconstructed image

c) The results of the reconstructed images after moving towards the cluster centre:

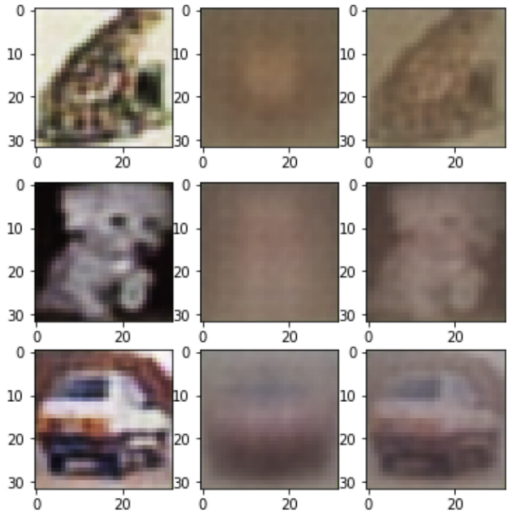


Fig. 7. Image of a class, Image of cluster center belonging to that class and reconstructed image

4) VAE

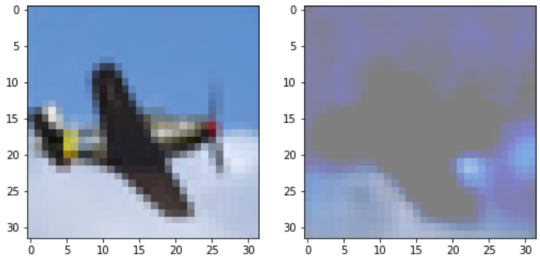


Fig. 8. Original Image and Reconstructed Image

The training of VAE for CIFAR-10 proved challenging for us. We tried a bunch of different models and changed a bunch of hyperparameters, and this was the best estimate we could get. We aren't comparing the VAE reconstructions with others because clearly the model fit isn't able to capture the image properly

B. Evaluation

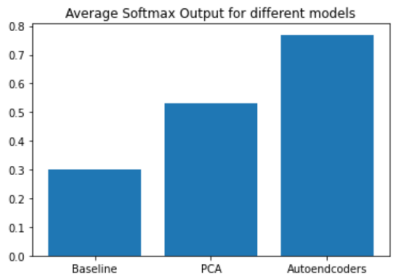


Fig. 9. Evaluation on Alexnet

For checking the quality of reconstruction of images which were generted by different models, we selected the predicted confidence threshold of a pretraianed classifier as an evaluation metric. This metric was chosen by us as a good reconstructed image would result in good confidence by the network and we could have used this reconstructed images as new training data for increasing the classifiers accuracy.

The results that we observed were expected as the baseline performed very bad but when the reconstructed images generated by AE were passed through a pretrained AlexNet, it gave us good confidence threshold of that class.

C. Hyperparamters

1) PCA
For choosing the number of hyperparameters for PCA, we wanted the model to be constructed such that the compoenents explain 95 percentage of variance of the data. Hence using the sklearn library functionalities, we chose the number of components as 217.

2) Autoencoders:

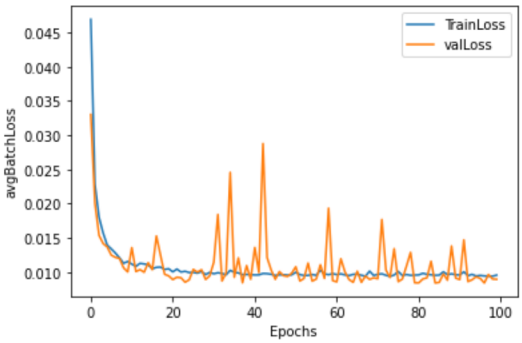


Fig. 10. Train Loss, Val Loss vs epochs

Hyperparameters	Value
Epochs	100
Learning rate	0.01
Criterion	MSE
Optimizer	Adam

3) Variational Autoencoders

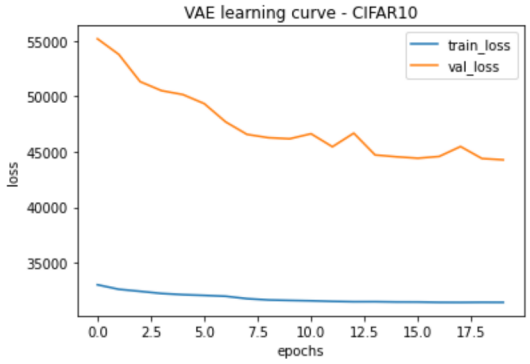


Fig. 11. Train Loss, Val Loss vs epochs

Hyperparameters	Value
Epochs	100
Learning rate	3e-4
Weight decay	1e-2
Criterion	ELBO + BCE
Optimizer	Adam

VII. Conclusion and Discussion

In this study, we focused on generating new images by learning the model of our given dataset. We started with a baseline average of the given images and moved out way up starting from a linear model (PCA) to non-linear models like auto-encoders and VAE's. For each model we perturbed the bottleneck dimensions in different ways and presented the best 3 methods out of them. We saw improving results as we went from our baseline to more complicated models like the autoencoders.

As PCA is a linear model, we were able to see it capture features for different classes, however, movement in the latent space for a linear model did not give us significantly good results which were expected. Moreover, we were able to find some interesting results in autoencoders. Particularly, we found synthetic image generation for the case of stochastic movement between two class' latent loadings as seen in Figure 6 (dog's image). Hence, we can say that autoencoders were actually able to capture non linearity in the image space for different classes.

To bolster our claim, we also wanted to include a classifier to evaluate the authenticity of the synthetic images along with a visual confirmation. For this, we also included an AlexNet classifier which gave us a degree of confidence for the generated images. Also, as expected, the average confidence of our synthetic image data was more in case of Autoencoders and PCA as compared to our baseline.

For future work, we would like to understand which direction in the latent space make sense if we are to attempt create believable reconstructions. We also intend to focus more to study the latent dimensions and how they cluster different classes together. Moreover, we also intend to study the density of these clusters along with how pure they are with respect to other similar classes. This would help us better understand the cluster movements and best direction to move in the latent space accordingly.

References

[Yang(2019)] Wei Yang. Classification on cifar-10/100 and imagenet with pytorch. https://github.com/bearpaw/pytorch-classification, 2019.

[Scikit-PCA(2013)] Scikit-PCA. Pca with sklearn. https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html, 2013.

[Bank et al.(2021)Bank, Koenigstein, and Giryes] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders, 2021.

[Chenjie(2019)] Chenjie. Building-autoencoders-in-pytorch. https://github.com/chenjie/PyTorch-CIFAR-10-autoencoder, 2019.

[Kingma and Welling(2014)] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2014.

[Malysheva(2018)] Sasha Malysheva. Vae with pytorch. https://github.com/SashaMalysheva/Pytorch-VAE, 2018.