Transfer Learning in β Variational Auto-Encoders

Alisa Yang
University of Toronto
alisayang@cs.toronto.edu

Val Kobilaski
University of Toronto
kobilaski@cs.toronto.edu

Abstract

In this paper, the potency of transfer learning is explored in an image reconstruction task. A novel ResNet18 based β-variational auto-encoder (β-VAE) architecture is proposed and developed. The architecture's performance is examined under separate training contexts. Namely, when the model is trained from random-initialization, when the encoder module of the model is initialized with pre-trained weights following training on an image classification task, and when the encoder is constructed with frozen pre-trained weight. An exploration of β-VAE specific hyper-parameters is also conducted. Further analysis is performed in order to observe the generative abilities of the models, as well as to observe the intermediate learned kernels. We find that a model trained from random-initialization outperforms a model pre-trained weights initialization in all metrics.

1 Introduction

In the field of deep learning, a common practice especially in sub-disciplines such as computer vision (CV) and natural language processing (NLP), is to take pre-trained models, and fine-tune the model to a new task by retraining the final forward layers. In CV for instance, a pre-trained ResNet18 model can be used, for any classification tasks, by keeping the learned parameters of the convolutional layers, and simply retraining the final classification layer.

Auto-encoders, and their subsequent variations namely, variational auto-encoders (VAEs) and β-variational auto-encoders (β-VAEs), have been well studied and produce excellent results in computer vision tasks, especially given their relatively simple concepts. In this paper an analysis of the effects of using a pre-trained model as the encoder module in a β-VAE architeture will be conducted.

To achieve this, a β-VAE architecture based on the ResNet18 CNN is constructed. An image reconstruction task will be formulated in order to evaluate model training time and model performance between models that utilize pre-trained ResNet18 parameters and models that train from random-initialization. An exploration of model hyper-parameters is also constructed, namely for finding viable values for the model's bottle-neck size, and β.

2 Related Works

The auto-encoder architecture originates as far back 1986 when it was proposed as a novel feed forward architecture by Rumelhart, Hinton, and McClelland [8]. The architecture is characterized by an encoder and decoder network module. The encoder network, takes the input and reduces the dimensionality to a small latent vector representation of the original input. This latent representation is then fed to the decoder module which reconstructs a higher dimensional representation of the latent vector (often to the size of the original input).

The auto-encoder architecture has seen wide application in the field of computer vision in the form of convolutional auto-encoders, which utilizes one or more convolutional layers. They have seen great success in a number of tasks including image de-noising[10], image compression and reconstruction[4], as well as image segmentation, classification, and many more.

The Variational Auto-Encoder (VAEs) has a similar architecture but with the added concept of variational inference. As with vanilla auto-encoders, the encoder aims to map the input data to a lower-dimension latent representation. However, with VAEs there's a fundamental assumption that the input datapoints are structured according to an underlying probability distribution. As such the latent representation takes the form of a a mean and standard of deviation to represent the datapoint in accordance to the underlying data distribution, and a stochastic element is added in the form of Gaussian noise. In addition, KL divergence is added to the loss function to act as a penalty term. This turns a regular auto-encoder, into a VAE, generative model. β-VAEs[3] are an extension to the idea, which introduces a learning

Preprint. Under review.

hyper-parameter: β. This parameter serves as a weight imposed on the KL-divergence score that is used as a loss term during training. β regulates the trade-off between the reconstruction and generative capabilities of the model.

ResNet18 is a CNN proposed by He et al.[2] in their foundational paper "Deep Residual Learning for Image Recognition". The model has gained significant attention in the field of computer vision for its state-of-the-art performance on image recognition tasks. ResNet18 is made all the more impressive due to it's relatively shallow architecture, containing approximately 11 million learning parameters (less than $\frac{1}{10}$ of VGG19's learning parameters). This is made possible due to its innovation; the skip connection, A forward network connection, that allows activations to surpass intermediate convolution modules. The skip connection is exceptionally effective at mitigating the vanishing gradient problem[5], common in most deep-learning architectures.

3 Methodology

3.1 Model Topology

β-VAEs consist of two modules, the encoder and the decoder. For the encoder module, a ResNet18 architecture was used. This model was chosen due to it's popularity and subsequent wide-availability of effective pre-trained classification models on a variety of datasets.

The encoder module consists of: An initial convolutional layer with a 7×7 kernel, followed by a max pooling layer, and a ReLU activation function. The network then contains four "residual blocks" as described in the original paper by He et al. [2]. Each residual block consists of the following layers: a 3×3 convolution, batch normalization, ReLU activation, a 3×3 convolution, batch normalization, down-sampling, and a final ReLU activation. These blocks increase in number of channels meanwhile decreasing in image width and height, by a factor of 2. Additionally, each block has a residual connection, originating from the input of the block to the output of the block. Following the four residual blocks, there is an average pooling layer, and then a final linear layer, of pre-specified size (denoted as z_{dim}, the size of this layer is tuned as a hyper-parameter).

As per the β-VAE algorithm, following the encoder module, the signal is propagated to two separate linear layers, the first linear layer will represent the mean (μ) of the Gaussian distribution that approximates the distribution of the latent representation of the input data, the second linear layer, represents the standard deviation (σ) of the distribution, as a measure of the distribution's "spread" in the latent space. A latent representation denoted as z, is then sampled from a fixed standard Gaussian distribution with ($\mu = 0, \sigma = 1$). The samples are then shifted by the previously encoded μ and scaled by the encoded σ. As stochastically sampling z is non-differentiable so we use the reparameterization trick introduced by [3], where μ and σ are deterministic and we introduce randomness through an error term ϵ sampled from the normal distribution. This lends to a stochastic latent space while still allowing for back-propogation.

The latent representation z, is finally fed through the decoder module. which is a version of the ResNet18 module, in which the layers are in reverse order. Additionally, all convolutional layers are replaced with transpose convolutional layers(with the same kernel size of 3×3), and all down-sampling (average and max pooling), are replaced by linear interpolation layers, which up-sample the intermediate image representation by a factor of two. Similar to the encoder, the decoder's residual blocks also contain residual connections directly connecting the input of a residual block to the output of the block. The model architecture can be visualized in figure 1

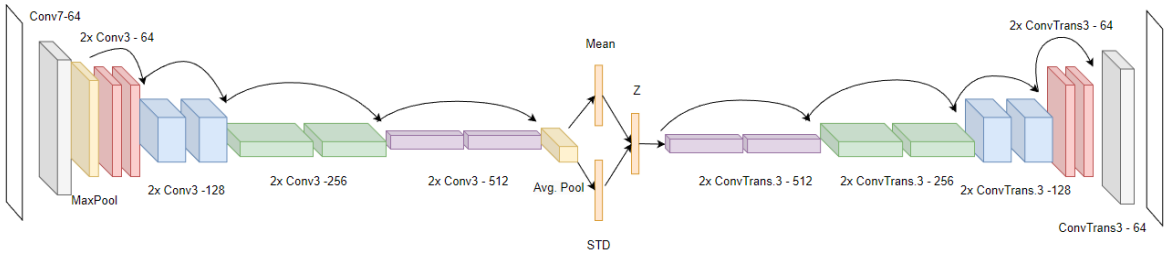


Figure 1: VAE architecture. ReLU and normalization layers omitted.

3.2 Model Training

The model was trained using the CIFAR-10 dataset[6]. The dataset consists of 60,000 images (32×32 RGB pixels) representing 10 separate image classes. The dataset was then split into a training, validation and test set (split of 10:1:1). Training images were normalized according to the published CIFAR-10 dataset mean and standard deviation pixel values.

Models were trained for 50 epochs with early stopping enabled (if a model did not improve it's validation loss for 5 consecutively epochs, training would terminate prematurely). The training loss, validation loss, and epoch duration in seconds, were saved for every epoch of training. This data was used for training analysis. The model's were trained using the Adam optimizer with an initial learning rate of $1e-3$.

The parameters of the network were optimized to reduce the composite β-VAE loss. This loss ($\mathcal{L}$) is a composite loss, consisting of the reconstruction loss ($\mathcal{L}_{recon.}$) and a KL-divergence loss ($\mathcal{L}_{KL}$) (figure 8). The reconstruction loss is a simple MSE loss between the original image x and the reconstructed image $\hat{x}$. The KL-divergence loss is used to encourage the latent representation to conform to the assumed normal distribution. A lower β will encourage reconstructed images to be visually closer to the original images, at the cost of generality when it comes to generating new images.

3.3 Experimentation and analysis

To evaluate the effect of using a pre-trained CNN as the encoder module, two variants of the model were trained and analyzed. The first variant was a 'control' model, in which both the encoder and decoder modules were trained from random-initialization. The second model will be the 'experimental' model, in which the encoder portion will be initialized from pre-trained weights, after it was trained on a CIFAR-10 classification task, the decoder module will still be trained from random-initialization. A third model was also proposed in which the encoder module, was initialized from the pre-trained weights, and then the gradients of the encoder were frozen, in order to retain the parameter values throughout the entirety of training.

Both models were also tested for their sensitivity to the mentioned hyper-parameters, namely different values were tested for both β and for z_{dim} (the dimension of the latent representation). The values tested were $[0, 1, 2]$ and $[10, 32, 64, 128]$ respectively. The training and validation losses were recorded throughout training as well as the epoch duration for performance and time analysis.

Following the training of the models, The models with the greatest training performance, were further analyzed. This was done, by visually evaluating the reconstructed images created by each model. The models were also evaluated on their ability to generate new images by giving a random sample with Gaussian distribution of the latent space to the decoder. The quality of the image will be assessed with the Laplacian kernel to estimate blurriness of the image [7]. Intermediate values were also analyzed through feature map visualization of intermediate convolutional layers.

4 Results and Analysis

A comparison of the image reconstruction outputs is provided in figure 4. The graphs represents the model's validation loss, for every epoch of training. A comparison is made based on the selected β value and the latent dimension size z_{dim}. the models with $z_{dim} = 128$, were able to achieve the lowest validation loss throughout training. However based on performance alone, there appears to be diminishing returns in expanding the latent dimension size, beyond the tested values. This is evident as the difference in performance between models with $z_{dim} = 128$ and $z_{dim} = 64$ is marginal when compared to the difference between models of $z_{dim} = 32$ and $z_{dim} = 10$. A visualization of the reconstructed images can be seen in figure 5.

A direct comparison of validation loss is made between the model trained from random-initialization (control model) and the model with the encoder module initialized with pre-trained weights (experimental model). Figure 2 depicts the validation loss between the control and experimental model with varying β values. From the data, the control model is able to achieve lower validation loss across all β values. This can primarily be explained by the fact that the encoders learned parameters in the experimental model are primarily tuned for classification. Possibly degrading the encoded representation as compared to the control model, which can learn a superior representation without any potentially biased convolutional kernels.

For training time, there is no significant difference between the control and experimental model. The control model took on average 142 seconds per epoch while the experimental model took 147 seconds per epoch. This trend was preserved through different (z_{dims}). This is because, although the number of nodes in the bottle-neck layer expand, the number of parameters in the encoder and decoder module greatly outnumber the parameters of the bottle-neck layer. A

model with $z_{dim} = 128$ has 23,910,275 parameters meanwhile a model with $z_{dim} = 64$ had 23,811,843 parameters. z_{dim} however did have a noticeable effect on the the number of epochs (variable due to early-stopping). Generally higher z_{dim} models ($z_{dim} = 128$) would take on average 40 epochs to converge, meanwhile models of $z_{dim} = 10$ took approximately 18 epochs to converge.

The model with the pre-trained encoder and weights frozen throughout training had abysmal performance. Image reconstruction outputs are available in figure 6. The reconstructed images reproduce the correct colour distribution for the image, however general shapes are severely blurred and unintelligible. Despite having roughly half the parameters to train, the model took approximately 121 seconds to train per epoch. The model trained for 29 epochs before being halted by early stopping for lack of improvement. For this reason the model was not evaluated further.

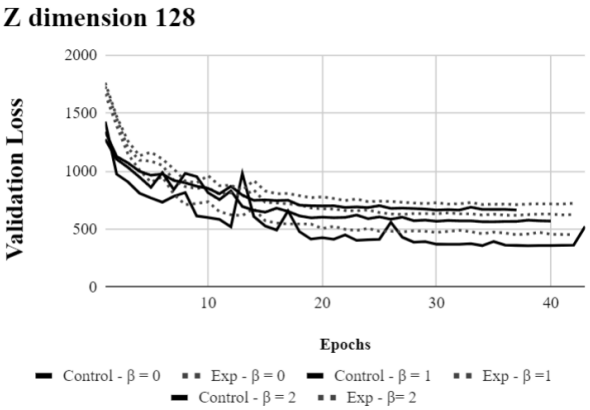


Figure 2: Notice that Control with the weights initialized from scratch is always lower than Experiment which is initialized from pre-trained weights.

The best performing model during training was the model trained from randomly-initialized weights, $z_{dim} = 128$, and $\beta = 0$, as demonstrated in figure 4. In terms of generative abilities however, the model's ability to generate inputs is greatly hindered, With $\beta = 0$, the model produces an almost 'tie-dye' like effect, as the model's features are severely entangled in the latent dimension. We see that increasing the value of β has a significant effect in the generation of new images, as seen in figure 3 conversely, increasing β increases validation loss. This is in-line with the findings of the original β-VAE paper [3]. This phenomenon showcases the fundamental trade-off between image reconstruction and disentanglement.

Supplementary analysis is also done using feature maps (available in figure 7). The feature mapping presents the activation at intermediate outputs at every layer. Four random kernels are chosen and their activations are plotted. From a visual inspection, the model trained from random-initialization retains a more consistent representation of the original image throughout the layers of the network. This is compared to the models with the encoder module training from pre-trained weights. This helps to validate the observation that the model trained from random-initialization performs the best throughout training.

For the quality of the image reconstruction, from visual inspection, we see that increasing β positively affects generation. $\beta = 2$ visually gives the best results, $\beta = 4$ has more artificial colors and is blurry. One measure for quality of image generation is the Laplacian variance of the image, the generated image is passed through a Laplacian kernel to evaluate the "blurriness" of the image. As seen in figure 9 there is a big discrepancy between the quality of generated images and reconstruction image for $\beta = 0$ and this gap decreases as we increase β. This is to be expected as higher variance mean sharper images, and corroborate our visual inspection of the results.

We do need to note that evaluation methods such as feature mapping and Laplacian variance are subjective and are just an additional evaluation method on top of visual inspection. A more sophisticated method is proposed by Leontev et al. [7]; A two stage classifier to measure the quality of the generated images is used. Due to the time constraint, we leave this as future work.

5 Conclusion

In this paper, it was found that the use of initializing pre-trained weights for the encoder module of a β- VAE does not provide an advantage over training from random-initialization, both in terms of training time and in model performance. In terms of hyper-parameters, the optimal z_{dim} was found to be 128, but it was only marginally better than $z_{dim} = 64$. It was found that the choice of β greatly influences the generative abilities of the model, with a higher β performing worse in the reconstruction task but better for image generation.

The topic would benefit from further analysis, ensuring the results are consistent when the experiment is replicated with similar CNN models such as VGG or AlexNet. Once replicated, the concept can additionally be extended to other variants of the VAE model such as VQ-VAEs[9] or even GAN-like systems[1]. Given additional computational resources the experiment would also benefit from utilizing larger and higher-resolution datasets such as ImageNet.

References

[1] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65, 2018.

[2] 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.

[3] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*, 2017.

[4] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[5] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[6] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[7] Mikhail Leontev, Alexander Mikheev, Kirill Sviatov, and Sergey Sukhov. Quality metrics of variational autoencoders. In *2020 International Conference on Information Technology and Nanotechnology (ITNT)*, pages 1–5, 2020.

[8] David E Rumelhart, Geoffrey E Hinton, James L McClelland, et al. A general framework for parallel distributed processing. *Parallel distributed processing: Explorations in the microstructure of cognition*, 1(45-76):26, 1986.

[9] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[10] Felix Weninger, Shinji Watanabe, Yuuki Tachioka, and Björn Schuller. Deep recurrent de-noising auto-encoder and blind de-reverberation for reverberated speech recognition. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4623–4627. IEEE, 2014.

Appendix

Github : Link to the Repository

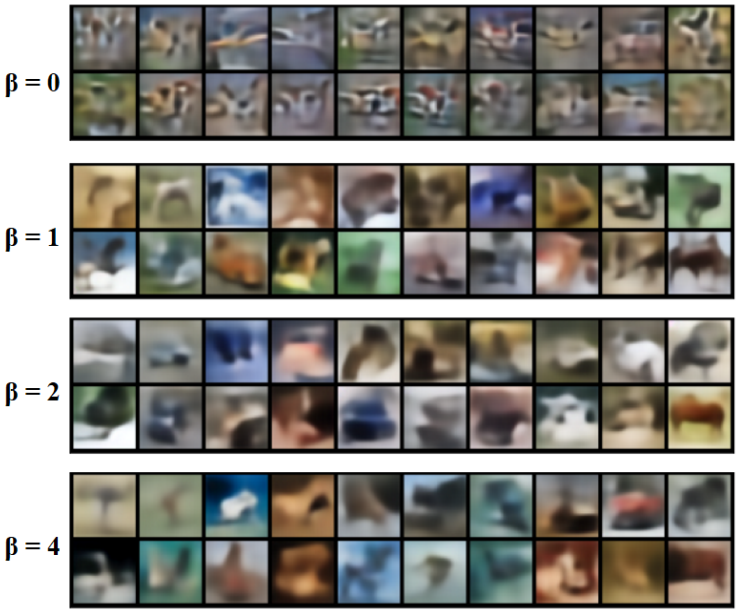


Figure 3: Generated images from random sample of the latent space ($z_{dim} = 128$). From top to bottom $\beta = 0, 1, 2, 4$. reconstruction quality is traded off for disentanglement.

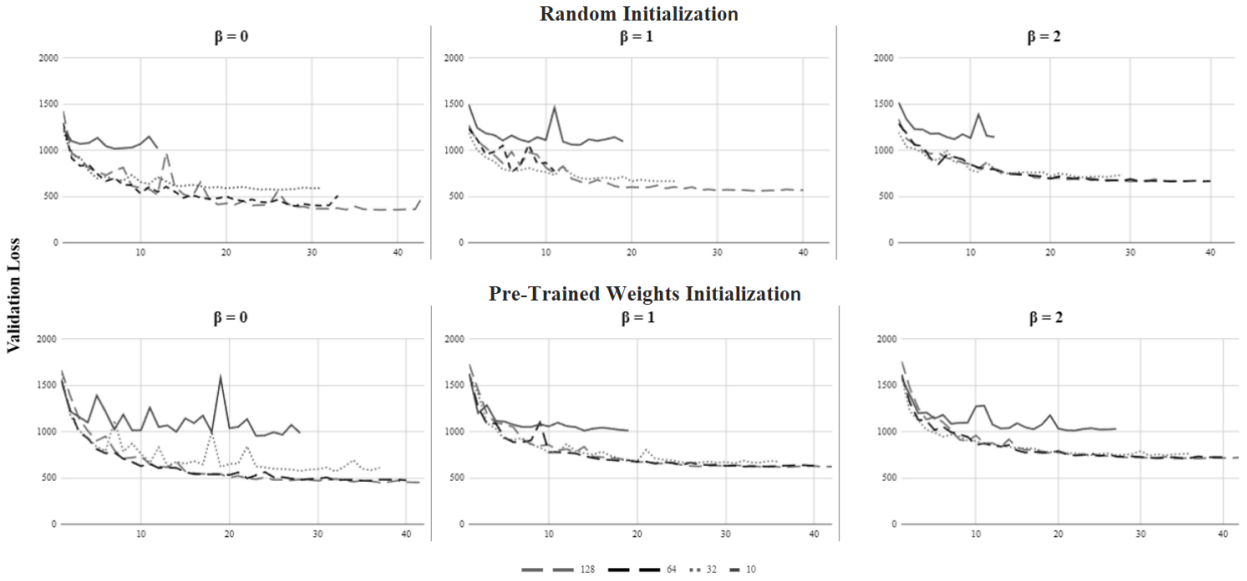
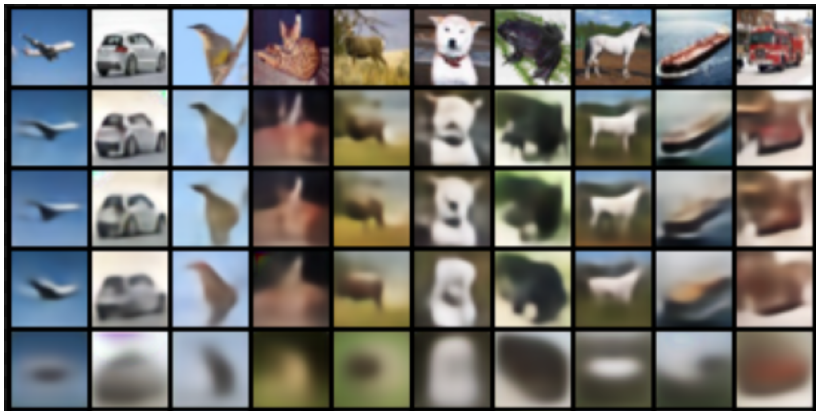
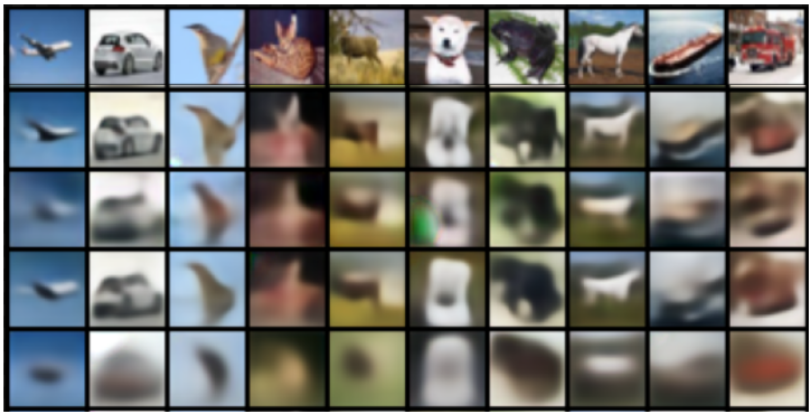


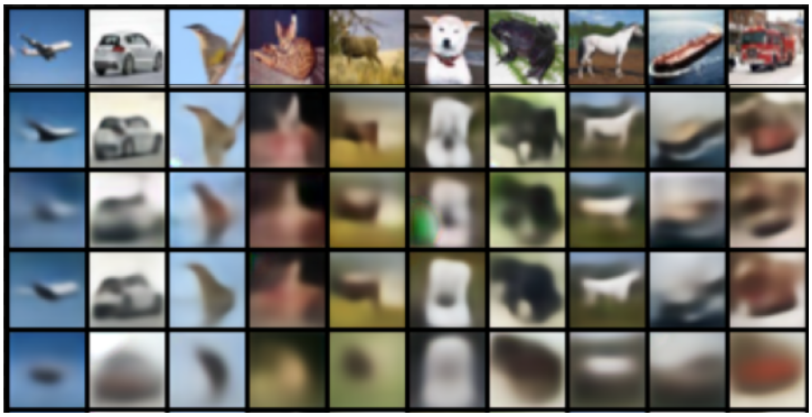
Figure 4: Validation loss for all models tested. Notable trends are: 1) Higher β shift the validation slope higher in the y-axis. 2) Latent dimension size of 10 gives the highest validation loss which suggest that 10 z_{dim} is too small of a bottle-neck to capture meaningful characteristics of the CIFAR10 classes. of size 64 adn 128 z_{dim} have comparable validation loss. 3) Random Initialization has lower validation score compared to Pre-Trained initialization for each β and z_{dim}.



(a) From top to bottom, original image, $z\_dim = 128$, $z\_dim = 64$, $z\_dim = 32$,$z\_dim = 10$, for model trained from scratch and $\beta = 0$.



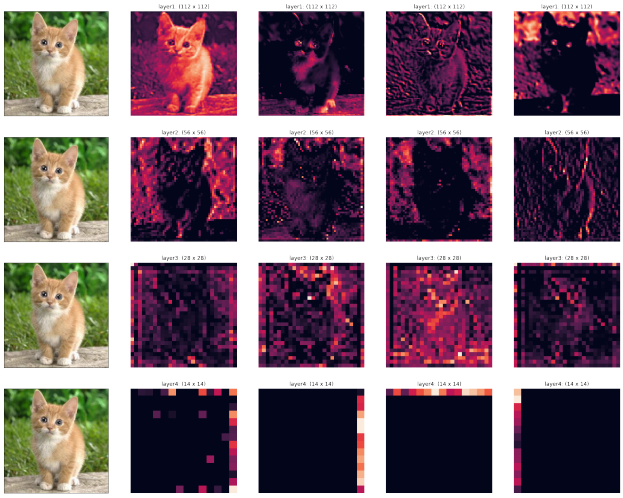
(b) From top to bottom, original image, $z\_dim = 128$, $z\_dim = 64$, $z\_dim = 32$,$z\_dim = 10$, for model trained from scratch and $\beta = 1$.



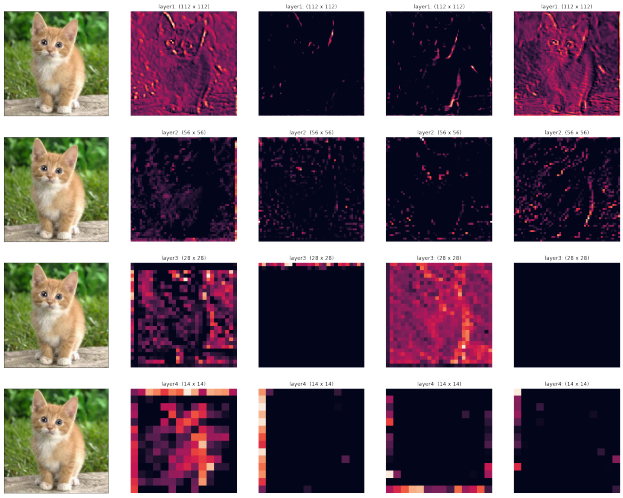
(c) From top to bottom, original image, $z\_dim = 128$, $z\_dim = 64$, $z\_dim = 32$,$z\_dim = 10$, for model trained from scratch and $\beta = 2$.

Figure 5: β and z_{dim} image reconstruction comparison for model trained from random initialization.

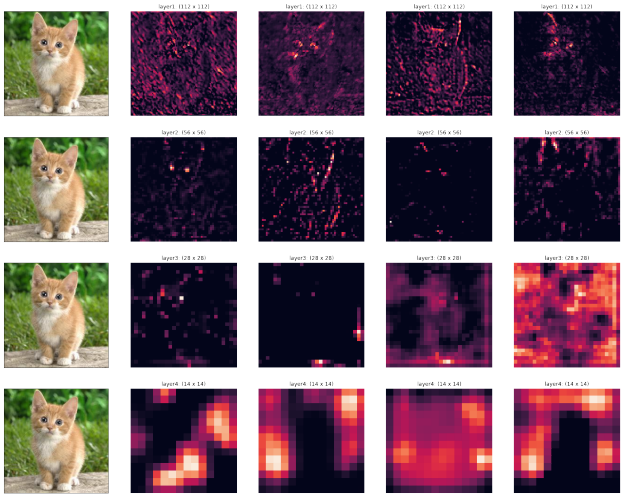
Figure 6: image reconstruction results for model with pre-trained encoder and frozen weights.



(a) Random-initialization model, feature map for encoder layers (in ascending order).



(b) pre-trained encoder model, feature map for encoder layers (in ascending order).



(c) pre-trained encoder model with frozen weights, feature map for encoder layers (in ascending order).

Figure 7: Feature maps for encoding module of the (a) random-initialization model, (b) pre-trained encoder model, and (c) pre-trained encoder model with frozen weights.

$$\mathcal{L} = \mathcal{L}_{recon.} + \beta \mathcal{L}_{KL}$$

$$\mathcal{L}_{recon.} = \sum_{i=1}^{BS=64} \|x_i - \hat{x}_i\|_2^2$$

$$\mathcal{L}_{KL} = -\frac{1}{2} \sum_{i=1}^{d} 1 + \log(\sigma_i^2) - \mu_i^2 - \sigma_i^2$$

Figure 8: Loss Functions for β-VAE

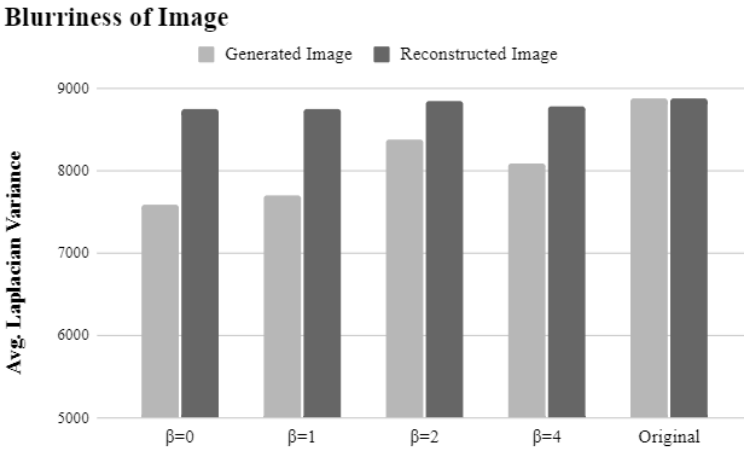


Figure 9: Average Laplacian Variance for five random images of both reconstructed and generated images. Note that as we increase β, we the reconstructed image is less blurry (more defined edges and shapes) but it does drop for $\beta = 4$. Blurriness of the pictures is by no means an perfect metric, but it an addition metric on top of visual inspection.

Contributions:

Pre-Writing

- Discussion and planing of the project (what to build, what to test, where to obtain the pre-trained model, dataset, etc.) - **Together**
- Encoder Module - **Alisa**
- Decoder Module - **Val**
- Loss function + VAE Module - **Together**
- Experiment Set up + Running test - **Alisa**
- Analysis of Results - **Together**

Writing of the Report

Abstract + Related Work + Methodology - **Val**

Introduction + Results - **Alisa**

Editing + Conclusion - **Together**

Resnet Architecture Graph + Feature Mapping Graph- **Val**

The rest of the graphs and images **Alisa**