VAE for Image Generation and Classification

Advanced Machine Learning (GR5242) Group Project

Zihan Zhou
M.A. Statistics
Columbia University
New York, NY 10027
zz2573@columbia.edu

Qianyu Zhao
M.A. Statistics
Columbia University
New York, NY 10027
qz2345@columbia.edu

Rener Zhang
M.A. Statistics
Columbia University
New York, NY 10027
rz2449@columbia.edu

Abstract

We designed and implemented two variational autoencoders (VAE) for the fashion-MNIST data, both of which are capable of generating artificial images that are similar to the original ones. We also evaluated the usefulness of this dimension reduction technique by training different classifiers on the hidden variables and achieved satisfying accuracy.

1 Introduction

Variational autoencoders (VAE) have been widely used for processing and modeling high-dimensional data and proved to be effective since its release in 2014 [1].In the field of neural network, people tend to treat the learning process as an extraction of the characteristics and dimension reduction. However, the question remains of how to sample from the known network. VAE solves the question. By utilizing the VAE, it is possible to generate samples based on the learned characteristics along with some easy-sample distribution. As a further study of VAE, we wish to evaluate the usefulness of the learned representations for a downstream classification task. This paper presents performances of different downstream classification choices(VAE + logistic regression / simple CNN / multiple CNN) on different sizes of the data regime and tries to figure out the best fit situations for each method. Moreover, we tried the VAE based on the continuous Bernoulli distribution, suggested by Loaiza-Ganem and Cunningham [2], instead of standard normal distribution.

The data set we are implementing our VAE for is the fashion-MNIST data. This data contains 70,000 grayscale images of apparels (e.g. shirts, trousers, bags etc.), each of which consists of $28 \times 28 \times 1 = 784$ pixels, and is classified into 10 classes.

2 Methods

2.1 VAE in General

In general, a VAE consists of three parts: an encoder, a decoder and a loss function. Suppose we have N data points $x_n \in \mathbb{R}^D$, where d can be enormously large. VAE provides with a method to reduce the dimension significantly by introducing a probability model that involves $X \in \mathbb{R}^D$ and a hidden variable $Z \in \mathbb{R}^H$, which has a certain prior. The encoder is a probability distribution that

Preprint. Under review.

approximates the true conditional distribution of $Z|X$, which is commonly chosen to be a Gaussian:

$$q_\phi(Z|X) = \prod_{n=1}^{N} q_\phi(Z_n|X_n), \text{where } q_\phi(Z_n|X_n) = \mathcal{N}(\mu_\phi(X_n), \text{diag}(\sigma_\phi^2(X_n))$$

Here $\mu_\phi(\cdot), \sigma_\phi^2(\cdot) : \mathbb{R}^d \to \mathbb{R}^h$ are neural networks parameterized by ϕ. Intuitively, q_ϕ is a distribution over Z values that are likely to produce a given X.

The decoder is the conditional distribution of $X|Z$, which approximates the marginal distribution of X:

$$X_n|Z_n \sim p_\theta(\cdot|z_n), Z_n \sim p_0(z), \text{for } n = 1, \ldots, N$$

where $p_0(z)$ is the given prior of Z and is conventionally chosen as standard Gaussian.

We then need to define an objective function to be optimized. Briefly speaking, we want the following two things [3]:

- Maximize the log likelihood $\log p(X)$;
- $q_\phi(Z|X)$ should be close enough to the true posterior $p(Z|X)$.

After some mathematical derivation, we obtained the following equation:

$$\log p(X) - KL(q_\phi(Z|X)\|p(Z|X)) = E_{Z\sim q_\phi}[\log p_\theta(X|Z)] - KL(q_\phi(Z|X)\|p_0(Z))$$

The regularizer, as shown in the function, is the Kullback-Leibler divergence between the distribution of the encoder and Z. The Kullback-Leibler divergence $KL(P(x)\|Q(x)) = E_{x\sim P}[\log P(x) - \log Q(x)]$ measures the similarity between the two distributions P and Q. Therefore the above equation gives us a valid objective function and the right hand side is defined as the evidence lower bound (ELBO). ELBO should be maximized over both generative and posterior parameters (ϕ, θ).

2.2 Normal VAE

As we mentioned above, the prior distribution for hidden variable Z, and the inference network, are mostly chosen to be Gaussian. The choice of generative network, i.e. $p_\theta(\cdot|z_n)$, is much more flexible in real practice.

One of the most commonly used family of distribution is Gaussian:

$$X_n|Z_n \sim \mathcal{N}(m_\theta(z_n), s_\theta^2(z_n)), Z_n \sim p_0(z) = \mathcal{N}(0, I), \text{for } n = 1, \ldots, N$$

Again $m_\theta(\cdot), s_\theta^2(\cdot) : \mathbb{R}^H \to \mathbb{R}^D$ are neural networks. The ELBO in this case can be written explicitly as

$$\begin{aligned} ELBO_{\mathcal{N}}(\theta, \phi) &= \sum_{n=1}^{N} E_{z_n\sim q_\phi}\left[\sum_{d=1}^{D} \log p_\theta(x_{nd}|z_n)\right] - KL(q_\phi\|\mathcal{N}(0, I)) \\ &= \sum_{n=1}^{N} E_{z_n\sim q_\phi}\left[\sum_{d=1}^{D} -\frac{(x_{nd} - m_{\theta,d}(z_n))^2}{2s_{\theta,d}^2(z_n)} - \frac{1}{2}\log(2\pi s_{\theta,d}^2(z_n))\right] - KL(q_\phi\|\mathcal{N}(0, I)) \end{aligned}$$

For a univariate normal $\mathcal{N}(\mu, \sigma^2)$, its KL-divergence with standard normal is

$$KL(\mathcal{N}(\mu, \sigma^2)\|\mathcal{N}(0, 1)) = \frac{1}{2}(\mu^2 + \sigma^2 - \log\sigma^2 - 1)$$

And the KL-divergence for multivariate normals is the summation of univariate KL-divergence over all dimensions.

2.3 Continuous Bernoulli VAE

Another commonly-used distribution for the decoder in previous examples is Bernoulli distribution:

$$X_n|Z_n \sim \mathcal{B}(\lambda_\theta(z_n)), Z_n \sim p_0(z) = \mathcal{N}(0, I), \text{for } n = 1, \dots, N$$

However, it is theoretically not suitable for [0,1] continuous variables, although we can still binarize the data and train such a VAE using Bernoulli likelihood. A solution that fixed this error is proposed this year, by introducing a continuous Bernoulli distribution [2], with density and mean

$$p(x|\lambda) = C(\lambda)\lambda^x(1-\lambda)^{1-x}, x \in [0,1], \text{where } C(\lambda) = \begin{cases} \frac{2\tanh^{-1}(1-2\lambda)}{1-2\lambda} & \text{if } \lambda \neq 0.5 \\ 2 & \text{otherwise} \end{cases}$$

$$\mu(\lambda) = \begin{cases} \frac{\lambda}{2\lambda-1} + \frac{1}{2\tanh^{-1}(1-2\lambda)} & \text{if } \lambda \neq 0.5 \\ 0.5 & \text{otherwise} \end{cases}$$

The continuous Bernoulli distribution is well-defined and works perfectly for modeling [0,1] continous variables, e.g. grayscale images. We can then define the continuous Bernoulli VAE analogously to the Bernoulli one:

$$X_n|Z_n \sim \mathcal{CB}(\lambda_\theta(z_n)), Z_n \sim p_0(z) = \mathcal{N}(0, I), \text{for } n = 1, \dots, N$$

The ELBO in this case is

$$ELBO_{\mathcal{CB}}(\theta, \phi) = \sum_{n=1}^{N} E_{z_n \sim q_\phi}\left[\sum_{d=1}^{D} x_{nd} \log \lambda_{\theta,d}(z_n) + (1 - x_{nd}) \log(1 - \lambda_{\theta,d}(z_n)) + \log C(\lambda_{\theta,d}(z_n))\right] - KL(q_\phi \| \mathcal{N}(0, I))$$

3 Implementation

For the Normal VAE, the encoder consists of one fully-connected layer with 128 units and ReLU activation, and an output layer with linear activations for the mean and the log of variance. The decoder takes in a $z \in \mathbb{R}^2 0$ and passes it through a fully connected layer with 128 units and sigmoid activation and an output layer with linear activation.

For the Continuous Bernoulli VAE, the encoder has a convolutional layer with 32 units, using a 3×3 filter and zero-padding to preserve image size, then a max-pooling layer with 2×2 window, and two fully connected layer with 512 units and ReLU activation each, followed by a dropout layer with parameter 0.1. The output layer has linear activation for the mean and softplus activation for the standard deviation. The decoder also has two fully connected layers with 512 units and ReLU activation and dropout layers, plus an output layer with sigmoid activation. Also we use taylor approximation in the calculation of the normalizing constant and mean of the continuous Bernoulli to avoid numerical issues, when λ is close to 0.5.

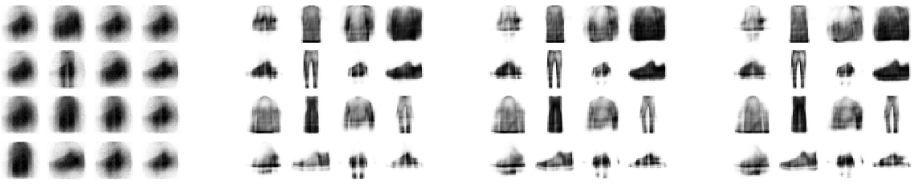
For both VAEs we set the latent dimension to be 20 and use Adam optimizer [4] with learning rate 0.001. We clipped the gradients in optimization step between -2 and 2 to improve numerical stability.

4 Results and Discussions

4.1 Sample Generation

One of the the biggest purposes of VAE is to generate artificial data. With the standard-normal-prior assumption, all we have to do is to sample from standard normal and feed it into the decoder.

For both models we train for 200 epochs. The training time for Normal VAE is around 8 seconds per epoch and 17 seconds per epoch for CB VAE, each using a training set of 60000 samples.



Epoch 1 Epoch 51 Epoch 101 Epoch 151

Figure 1: Samples from Normal VAE

Epoch 1 Epoch 51 Epoch 101 Epoch 151

Figure 2: Samples from $\mathcal{CB}$ VAE

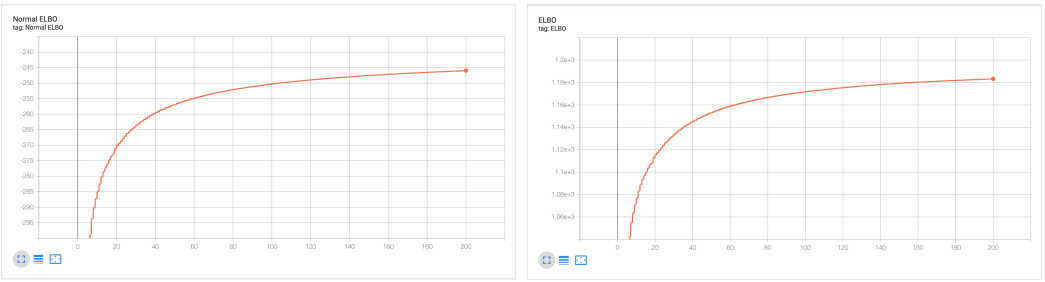


Figure 3: Normal ELBO & $\mathcal{CB}$ ELBO

The generated samples became less blurry after several epochs. They are generally consistent with the change of ELBO's over epochs: both the ELBO of normal and $\mathcal{CB}$ converges roughly after the 100^{th} epoch, and the generated samples do not change much after then.

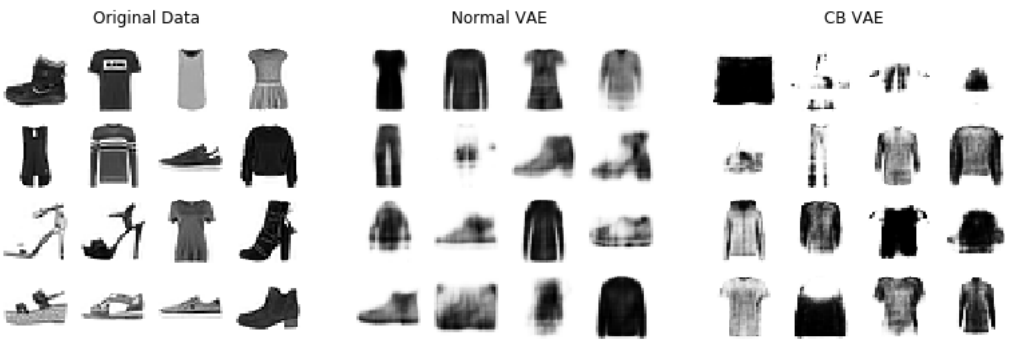


Figure 4: Original and Artificial Samples

We can easily identify most of the sampled pictures with some exceptions which seem to be a weird combination of two categories. Overall, we believed that the trained VAE is efficient and valid. It

seems that Normal VAE generate more blurry samples while samples generated by CB VAE are sharper.

4.2 Sample Reconstruction

We can also reconstruct a sample by feeding it into the inference network to obtain the corresponding hidden variable, and feed it back into the generative network. We achieved satisfying reconstruction quality for both models. The characteristics still hold: samples from Normal VAE are more blurry while samples from $\mathcal{CB}$ VAE are close to binarized.

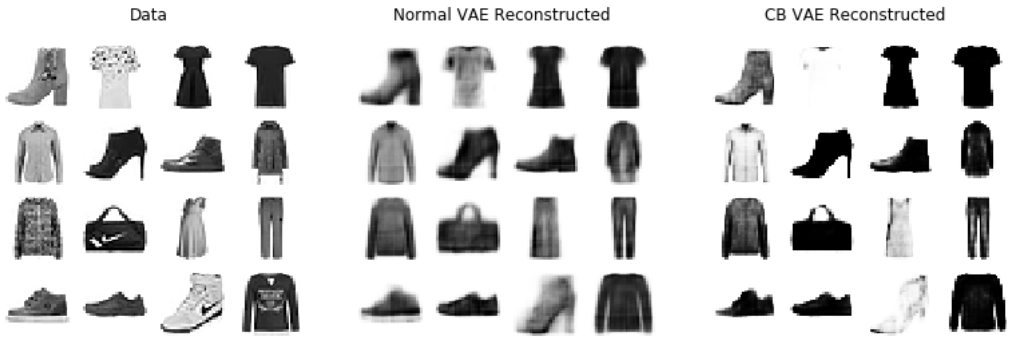


Figure 5: Original and Artificial Samples

4.3 Classification Using VAE

As we mentioned in the introduction, VAE can be seen as a dimension reduction technique: the hidden variables are essentially representations of original variables in a significantly lower dimension (in our experiment $784 \to 20$). Therefore it is natural to check how well these hidden variables extract the characteristics from original ones by doing classification.

The experiments are conducted repeatedly in two regimes: one using a larger subset of data as training set and the held-out data as testing set, the other one switching the two. We use the original training set of size 60000 as the larger subset, and the testing set of size 10000 as the smaller one.

Three classifiers are trained separately: logistic regression, simple CNN and multilayer CNN. The simple CNN classifier consists of one convolutional layer (with 32 units, 3×3 filter and size-preserving zero-padding) and one fully connected layer with softmax activation. The multilayer CNN classifier has the same convolutional layer followed by a max-pooling layer with 2×2 filter, then a fully connected layer with 128 units and ReLU activation, and an output layer with the same structure as simple CNN.

The test results are shown in Table 1. Further details are attached in the appendix.

Table 1: Test Accuracy

	Logistic	Simple CNN	Multilayer CNN
Original Data (Large)	0.8386	0.8364	0.9071
Original Data (Small)	0.8251	0.8269	0.8767
$\mathcal{N}$-VAE (Large)	0.7459	0.7484	0.7914
$\mathcal{N}$-VAE (Small)	0.7028	0.7523	0.7569
$\mathcal{CB}$-VAE (Large)	0.7774	0.7826	0.8223
$\mathcal{CB}$-VAE (Small)	0.7271	0.7805	0.7844

The results show that our VAEs generate hidden variables that successfully extract most of the information from the original data: the classification accuracy using hidden variables is still at a

satisfying level, compared to using original data. The $\mathcal{CB}$ VAE has better performances than the normal VAE in all three scenarios, leading the accuracy by roughly 3%.

In terms of different data regimes, the dimensionality reduction from the VAE makes simple CNN classifier more robust to the size of training set, than the multilayer CNN. The accuracy of multilayer CNN drops sharply when switching to a small training set, while simple CNN has an unchanged, ever slightly better accuracy. It implies that for a complex classifier, its accuracy depends highly on the size of training set, since a small training set may not be enough for it to generalize well. For a simple classifier, it might be more advantageous when we only have a small number of samples for training, and when we are more concerned about computing efficiency rather than accuracy using only a representation of data in a lower dimension.

5 Conclusions

In this project we implement two types of VAEs for grayscale images, including one that is newly proposed. We compare their capabilities of generating artificial images and image reconstruction. We also evaluate the usefulness of the learned representations by training several different classifiers on them and suggest that the VAE might be helpful for classification when training data is inadequate and of low dimension.

References

[1] D. P. Kingma & M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

[2] Gabriel Loaiza-Ganem &. John P. Cunningham. The continuous Bernoulli: fixing a pervasive error in variational autoencoders. In *Advances in Neural Information Processing Systems, 13266-13276*, 2019.

[3] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[4] D. P. Kingma & J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[5] Convolutional Variational Autoencoder: `https://www.tensorflow.org/tutorials/generative/cvae`.

Appendix

The appendix contains details on the training process of classifiers.

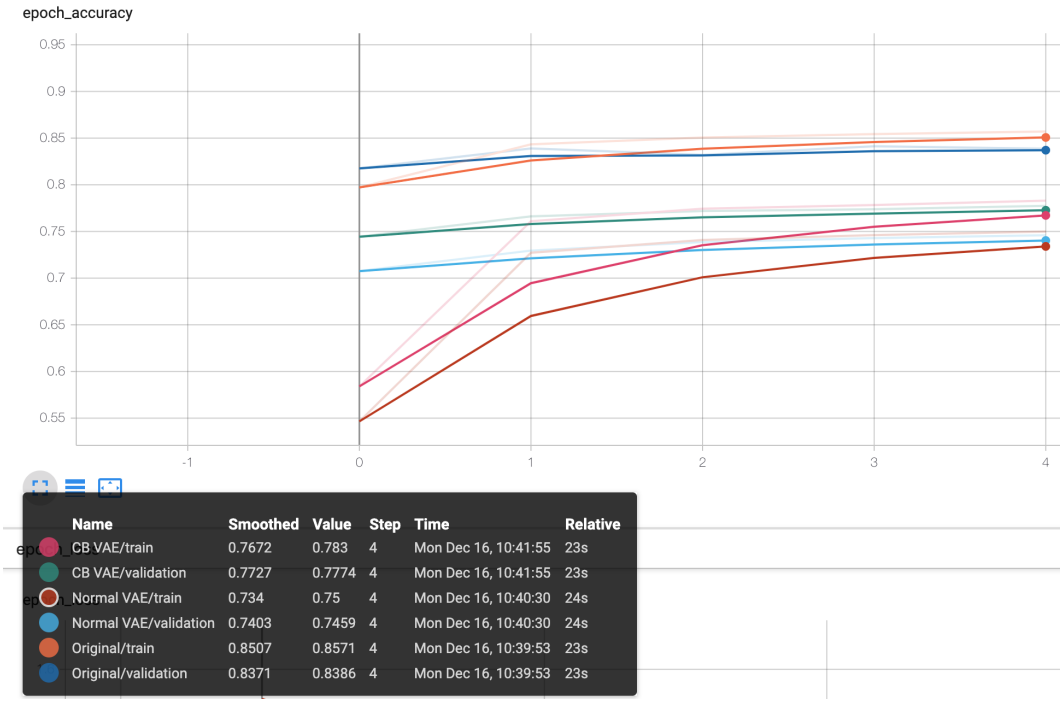


Figure 6: Logistic on Large Training Set

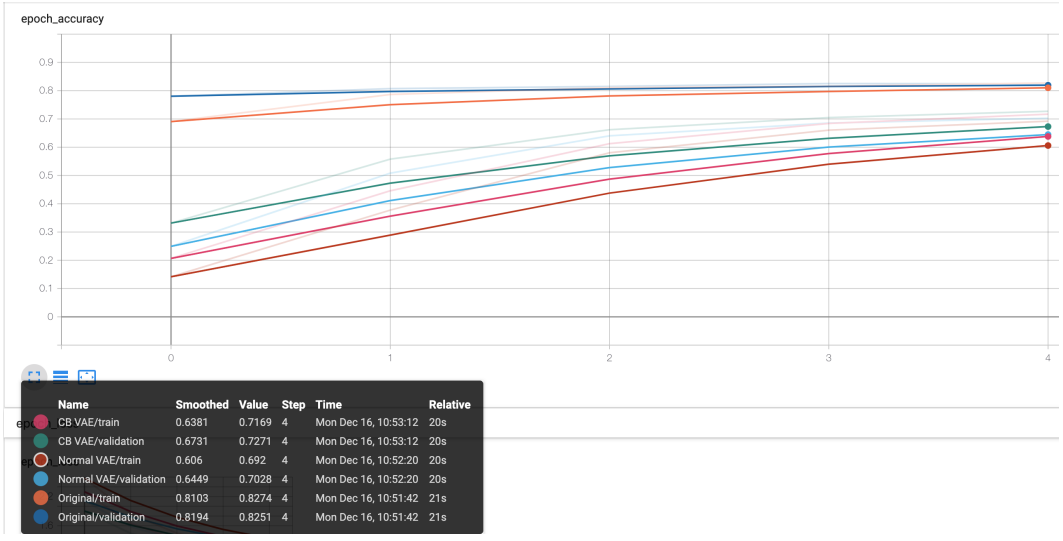


Figure 7: Logistic on Small Training Set

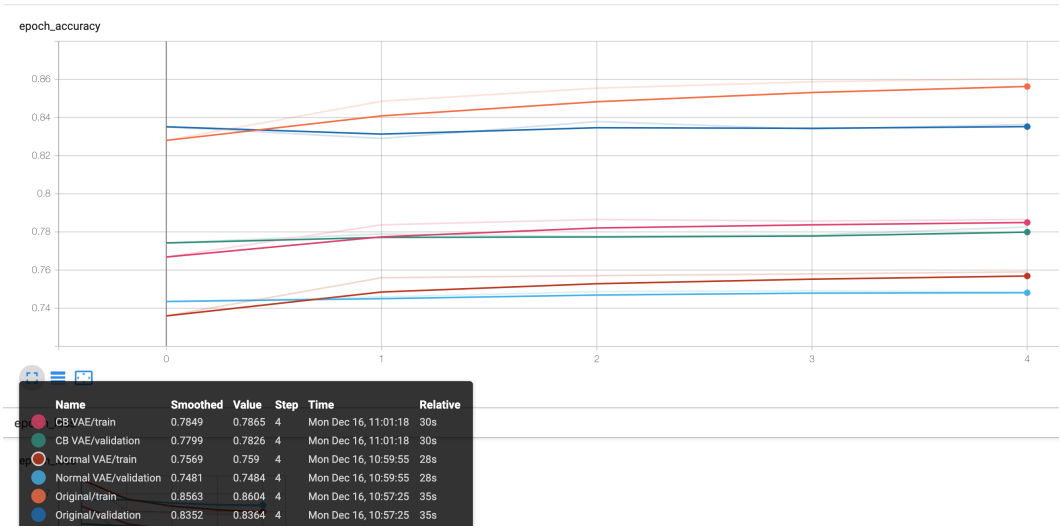


Figure 8: Simple CNN on Large Training Set

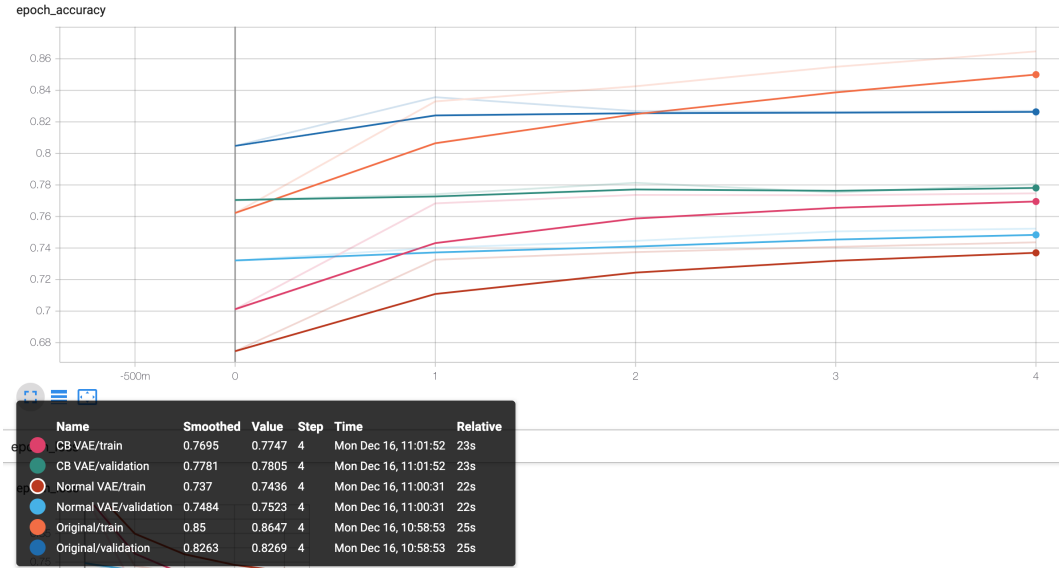


Figure 9: Simple CNN on Small Training Set

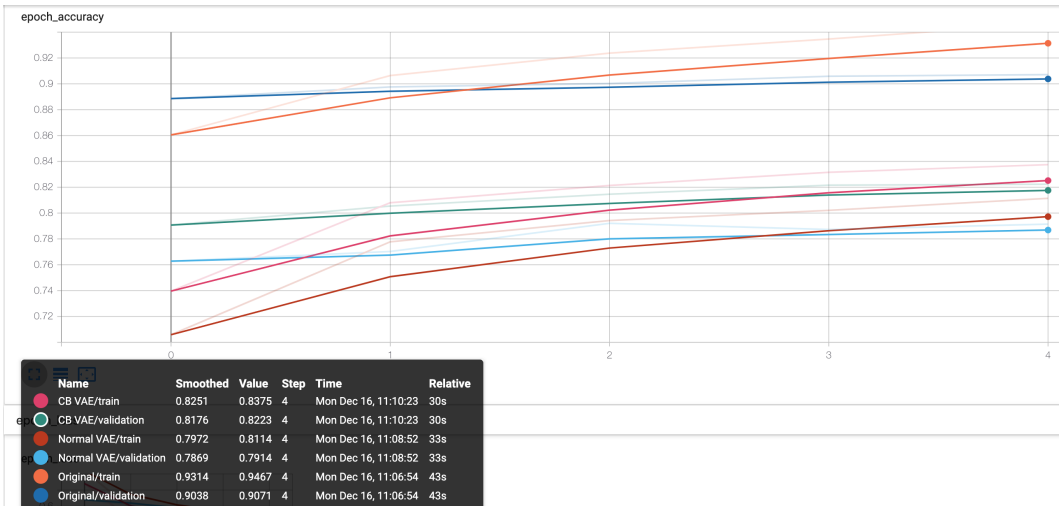


Figure 10: Multilayer CNN on Large Training Set

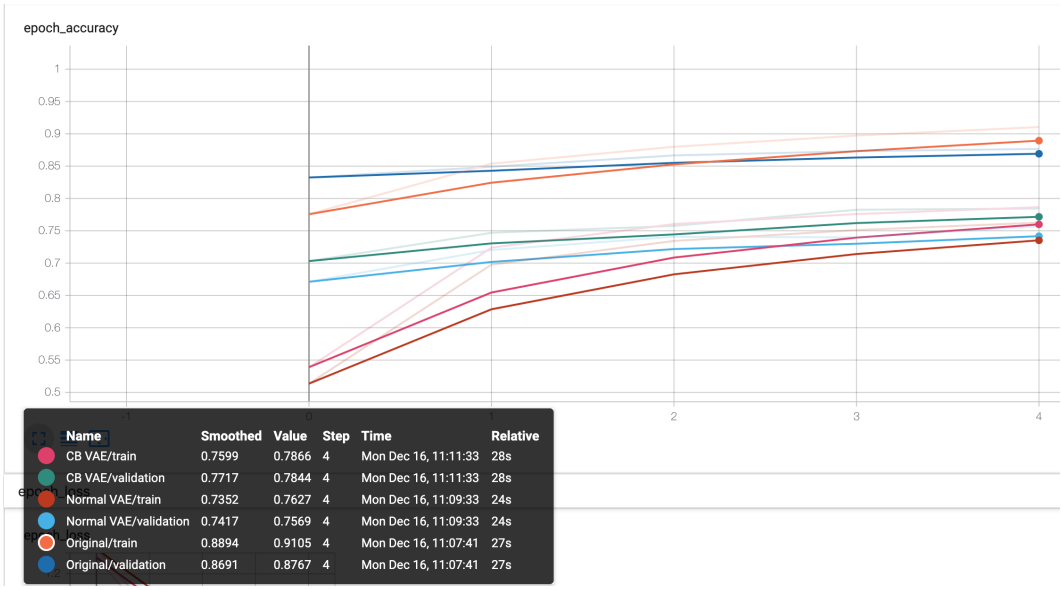


Figure 11: Multilayer CNN on Small Training Set