Advanced Topics in ML: Proposal

Riccardo Giacometti, Daniil Terekhin, Timur Taepov, Baris Aksakal

October 2023

Contents

1 Introduction

In this proposal, we aim to achieve state of the art performance in image classification for the CIFAR-10 and CIFAR-100 datasets. The authors propose a modified architecture of Stacked What Where Autoencoders where the latent space is made variational through the use of Dirichlet variational autoencoders (DirVAE) due to their significant performance in creating a better latent space representation which assists in image classification tasks. In addition, β-VAEs are also exploited for the same objective of better latent representation. This modified architecture called β-SWWDirVAE will then be attached to EfficientNet, a state of the art network, and optimised with Sharpness Awareness Minimisation (SAM) which is hypothesised to yield superior results.

2 Preliminaries

2.1 Stacked What Where Autoencoder (SWWAE)

The SWWAE architecture was proposed by Zhao, J. et al. (2015) which improves on stacked autoencoders by integrating discriminative and generative pathways and provides a unified approach to supervised, semi-supervised and unsupervised learning without relying on sampling during training such as in Deep Boltzmann Machines (DBM) where the sampling is performed in every layer which is computationally expensive to train. In addition, SWWAEs are less time consuming to train in comparison to traditional stacked autoencoders because stacked autoencoders need to be pre-trained layer by layer which then need to be fine tuned.

SWWAEs are comprised of a convolutional network that acts as the encoder and a deconvolutional network which acts as a decoder. The "what" and "where" of the model are variables give the network more information regarding the feature and its position in the input data. The "what" variable is the output of max-pooling layers, representing content without complete positional information. Conversely, the "where" variable, derived from the max-pooling switch positions, holds the positional information required for reconstruction. The model employs a feed-forward Convnet and a feedback Deconvnet, where the "what" variables are used for forward computation, and the "where" variables aid in accurately reconstructing the input during the backward pass, essentially informing where dominant features are located in the input data.

The problem with this so far is that the max pooling layers are not differentiable with respect to the argmax which are where the "where" variable lies. Therefore, the paper also mentions work from Goroshin et al. (2015) which created a "soft" version of the max and argmax pooling operators (Equations 1 and 2).

$$m_k = \sum_{N_k} z(x,y) \frac{e^{\beta z(x,y)}}{\sum_{N_k} e^{\beta z(x,y)}} \approx \max_{N_k} z(x,y) \tag{1}$$

$$\mathbf{p_k} = \sum_{N_k} \begin{bmatrix} x \\ y \end{bmatrix} \frac{e^{\beta z(x,y)}}{\sum_{N_k} e^{\beta z(x,y)}} \approx \arg\max_{N_k} z(x,y) \tag{2}$$

Where β is a hyper parameter where smaller values emulate mean pooling and higher values emulate max pooling. $z(x,y)$ denotes activation on the feature maps i.e the output, and x, y represents the spatial location which take normalized values from -1 to 1.
In addition to the "what" and "where" variables, L2 regularization is implemented to corresponding encoder and decoder layers i.e the i^{th} layer in the encoder is L2 normalised with the i^{th} layer in the decoder. These regularizers are used because they ensure a structured alignment between feature planes between the "where" map from the i^{th} layer in the encoder and the "what" from

the i^{th} layer in the decoder. This alignment is crucial as it prevents any disorder that could impair the function of the unpooling step. Secondly, when training with classification loss, these intermediate terms prevent a situation where only the lower layers are active in reconstruction while the upper layers remain idle. This is important as it ensures that filters from all layers are utilized and regularized. Lastly, the text draws a parallel between this methodology and layer-wise auto-encoder training. In the SWWAE, each segment comprising intermediate encoder, pool, unpool, and decoder units, paired with the L2 terms, functions similarly to a single-layer convolutional auto-encoder.

Given these terms, the loss function for this model can be seen in Equation 3.

$$L = L_{NLL} + \lambda_{L2rec} L_{L2rec} + \lambda_{L2M} L_{L2M} \tag{3}$$

Where L_{L2rec} is the reconstruction loss between the input and output of the model along with a hyperparameter weight λ_{L2rec}. The L_{L2M} is the intermediary L2 normalisation as explained before which also has a hyperparameter weight λ_{L2M}. The L_{NLL} loss is the negative log-likelihood that is used when this model is in its supervised learning mode, in the case of unsupervised learning this term becomes omitted from the loss function. A visualisation of the "what" and "where" concepts along with the overall architecture are shown in Figure 1 (a) and (b), respectively.

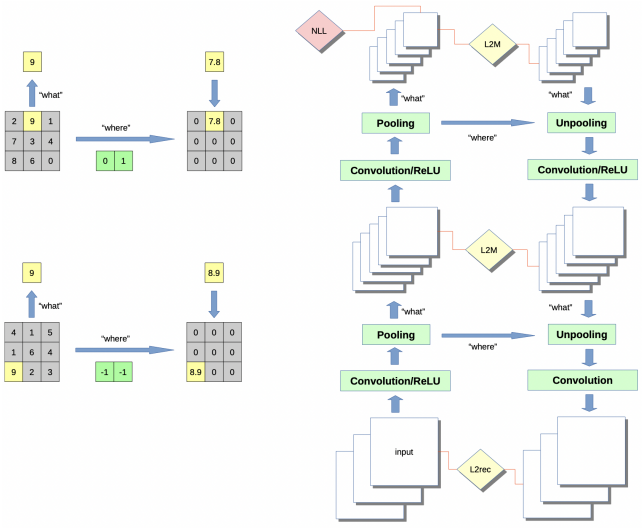


Figure 1: Left (a): pooling-unpooling using "what" and "where" concepts. Right (b): model architecture where dense layers are omitted for simplicity. (Zhao, J. et al., 2015)

The results from CIFAR-10 and CIFAR-100 datasets in comparison to other

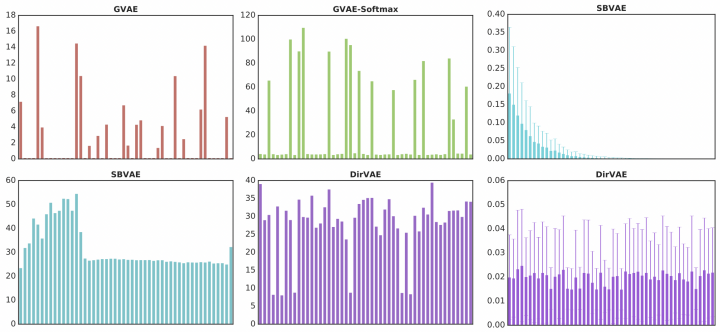
models around the same time are shown in Table 1. The way this model was trained was by performing unsupervised learning on the 80 million tiny images dataset which the CIFAR datasets are derived from and then the encoder was attached to a VGG-style network that was then trained on the CIFAR datasets. It can be seen that the accuracy for the CIFAR-10 dataset was very close to the performance of the best model (All-Covnet) and as for the CIFAR-100 dataset it outperformed all models.

Table 1: Results from SWWAE on CIFAR-10 and CIFAR-100 (Zhao, J. et al., 2015)

model	CIFAR-10	CIFAR-100
All-Convnet (Springenberg et al. (2014))	**92.75**%	66.29%
Highway Network (Srivastava et al. (2015))	92.40%	67.76%
Deeply-supervised nets (Lee et al. (2014))	92.03%	65.43%
Fractional Max-pooling with large augmentation (Graham (2014))	95.50%	68.55%
SWWAE ($\lambda_{L2rec} = 1, \lambda_{L2M} = 0.2$)	92.23%	**69.12**%
Convnet of same configuration	91.33%	67.50%

2.2 Dirichlet VAE (DirVAE)

The DirVAE was proposed by Joo, W. et al. (2019) and involves replacing the continuous latent variables modelled by a Gaussian distribution with discrete latent variables that follow a Dirichlet distribution which is composed of Gamma random variables. In DirVAE, the inverse Gamma CDF is approximated using an approximation function. The main findings behind this is that a Dirichlet distribution is better at representing categorical data because it can capture multi-modality which traditional VAEs cannot. In addition, this paper discusses the issue of component collapsing where a substantial amount of decoder weights from the latent neurons to the subsequent decoder neurons are near-zero. When these weights approach zero, the impact of the latent dimensions on the next decoder diminishes, leading to inefficient learning within the given neural network framework. The paper also investigates the other extreme known as latent value collapsing where network tends to find solutions where a portion of the latent variables do not carry any meaningful information about the data. This can lead to a collapse of the latent space where many dimensions become inactive or redundant. From this paper, DirVAEs have shown to be very robust from these two problems as show in Figure 2.



(a) Latent dimension-wise L2-norm of decoder weights of VAEs. (b) Latent values of VAEs.

Figure 2: Latent and component value collapsing (Joo, W. et al. 2019)

In addition to the reasons given above, this paper also shows that DirVAEs have increased interpretability of the latent space where Figure 3 shows the latent space representation of the MNIST dataset using DirVAE in comparison to a simple t-SNE plot and traditional Gaussian VAEs (GVAE). It can be seen that the DirVAE has much better separation and cohesion in comparison to the other two methods.

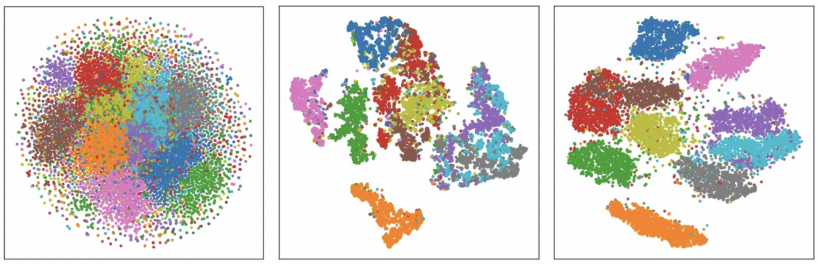


Figure 3: Latent representation of t-SNE (left), GVAE (middle) and DirVAE (right) (Joo, W. et al. 2019)

What is even more interesting, are the results from using the latent space from the DirVAE and feed it to a nearest neighbours algorithm (kNN) where the results are summarised in Table 2.

Table 2: The error rate of kNN with the latent representations of VAEs.(Joo, W. et al. 2019)

	MNIST ($K = 50$)			OMNIGLOT ($K = 100$)		
	$k = 3$	$k = 5$	$k = 10$	$k = 3$	$k = 5$	$k = 10$
GVAE (Nalisnick et al., 2016)	28.40	20.96	15.33	–	–	–
SBVAE (Nalisnick et al., 2016)	9.34	8.65	8.90	–	–	–
DLGMM (Nalisnick et al., 2016)	9.14	8.38	8.42	–	–	–
GVAE	$27.16_{\pm 0.48}$	$20.20_{\pm 0.93}$	$14.89_{\pm 0.40}$	$92.34_{\pm 0.25}$	$91.21_{\pm 0.18}$	$88.79_{\pm 0.35}$
GVAE-Softmax	$25.68_{\pm 2.64}$	$21.79_{\pm 2.17}$	$18.75_{\pm 2.06}$	$94.76_{\pm 0.20}$	$94.22_{\pm 0.37}$	$92.98_{\pm 0.42}$
SBVAE	$10.01_{\pm 0.52}$	$9.58_{\pm 0.47}$	$9.39_{\pm 0.54}$	$86.90_{\pm 0.82}$	$85.10_{\pm 0.89}$	$82.96_{\pm 0.64}$
DirVAE	$\mathbf{5.98_{\pm 0.06}}$	$\mathbf{5.29_{\pm 0.06}}$	$\mathbf{5.06_{\pm 0.06}}$	$\mathbf{76.55_{\pm 0.23}}$	$\mathbf{73.81_{\pm 0.29}}$	$\mathbf{70.95_{\pm 0.29}}$
Raw Data	3.00	3.21	3.44	69.94	69.41	70.10

As can be seen, the DirVAE outperforms all of the compared models across all parameters for k in both datasets tested with special mention to the performance for the MNIST where the model outperforms traditional VAEs by factor of roughly 4 to 5-fold.

As mentioned in the paper, formulating the probabilistic process of the model involves defining the generative sub-model which is responsible for generating data given a set of latent variables. The key distinction between the Dirichlet Variational Autoencoder (DirVAE) and Gaussian Variational Autoencoder (GVAE) lies in the prior distribution assumption on the latent variable z. In DirVAE, a Dirichlet distribution is employed instead of the standard Gaussian distribution used in GVAE. Which can be represented by Equation 4.

$$\mathbf{z} \sim p(\mathbf{z}) = Dirichlet(\boldsymbol{\alpha}), \mathbf{x} \sim p_\theta(\mathbf{x}|\mathbf{z}) \tag{4}$$

Where α is a parameter of the Dirichlet distribution, z is the latent variable, x is the generated data, and θ are the parameters of the generative model. In addition, this model entails a probabilistic encoder with an approximating posterior distribution $q_\theta(z|x)$ which is designed to be $Dirichlet(\hat{\alpha})$ Here, $\hat{\alpha}$ is derived from the observation x through a Multi-Layer Perceptron (MLP) with a softplus output function ensuring positive values constrained by the Dirichlet distribution. Instead of sampling $\mathbf{z}$ directly from the Dirichlet distribution, the Gamma composition method is used which is explained in Section 2.2 of the paper. Initially, $\mathbf{v}$ is drawn from a MultiGamma distribution as shown in Equation 5 which was derived from Appendix B of the paper.

$$\mathbf{v} \sim MultiGamma(\boldsymbol{\alpha}, \beta \cdot \mathbf{1}_K) \tag{5}$$

Where $\alpha_k, \beta > 1$ for $k = 1, ..., K$. This is the normalised using its summation $\sum v_i$ The objective function to optimize the model parameters θ and ϕ is then composed. The following equations then lead to the final loss function loss function defined in Equation 6.

$$L(x) = \mathbb{E}_{q_\phi(x|z)}[logp_\theta(x|z)] - KL(P||Q)$$

$$P = MultiGamma(\boldsymbol{\alpha}, \beta \cdot \mathbf{1}_K)$$
$$Q = MultiGamma(\hat{\boldsymbol{\alpha}}, \beta \cdot \mathbf{1}_K)$$
$$KL(P||Q) = \sum log\Gamma(\alpha_k) - \sum log(\hat{\alpha}_k) + \sum(\hat{\alpha}_k - \alpha_k)\psi(\alpha_k)$$

$$L(x) = \mathbb{E}_{q_\phi(x|z)}[logp_\theta(x|z)] - (\sum log\Gamma(\alpha_k) - \sum log(\hat{\alpha}_k) + \sum(\hat{\alpha}_k - \alpha_k)\psi(\alpha_k)) \tag{6}$$

Where k is the k^{th} gamma random variable in the latent space vector and ψ is the digamma function which is the logarithmic derivative of the gamma function.

In order to approximate the Gamma random variables which compose the Dirichlet distribution, the paper uses the inverse Gamma cumulative distribution function (CDF) which is shown in Equation 7.

$$F^{-1}(u; \alpha, \beta) \approx \beta^{-1}(u\alpha\Gamma(\alpha))^{\frac{1}{\alpha}} \tag{7}$$

Hence, and auxiliary variable $u \sim Uniform(0, 1)$ to take over the randomness of the random variable X. The Gamma-sampled X is treated as a deterministic value in terms of α and β.

2.3 β-VAE

Compared to the last sections, this is a simple yet sophisticated modification to VAEs. Higgins, I. et al. (2017) proposed in adding a parameter, β, for manipulating the trade off between reconstruction error and posterior distribution approximation (KL Divergence). The β parameter is introduced as an adjustable multiplier for the KL Divergence loss in the overall loss function, this is summarised in Equation 8.

$$\mathcal{L}(\theta, \phi; \mathbf{x}, \mathbf{z}, \beta) = \mathbb{E}_{q_\phi(x|z)}[logp_\theta(\mathbf{x}|\mathbf{z})] - \beta D_{KL}(q_\theta(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) \tag{8}$$

This loss function facilitates the learning of more "clean" and disentangled latent representations compared to previous methods like InfoGAN and DC-IGN without requiring design decisions or assumptions about the data as shown in the paper. The authors of the paper demonstrate through their research that when the β hyperparameter is tuned appropriately ($\beta > 1$), the β-VAE outperforms the standard VAE ($\beta = 1$), as well as state-of-the-art unsupervised (InfoGAN) and semi-supervised (DC-IGN) approaches to disentangled factor learning across several datasets like celebA, faces, and chairs datasets. Essentially, this paper tries to prioritise a better latent space rather that reconstruction capabilities.

2.4 Sharpness-Aware Minimisation (SAM)

Sharpness-Aware Minimization (SAM) (Foret, P. et al., 2020) is second order optimisation algorithm that not only minimises the loss value but also sharpness of the computational graph. The additional benefits to this algorithm is that it is efficient because it captures curvature without having to calculate the second order derivative. SAM is also shown to improve generalisation and is robust against image noise. A representation of the computational graph comparing stochastic gradient descent (SGD) against SAM and SGD is shown in Figure 4.

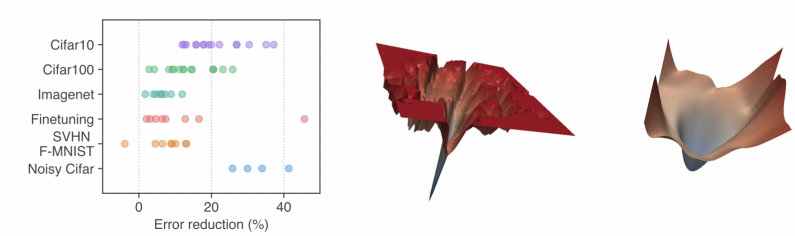


Figure 4: (a) Error rate reduction obtained by switching to SAM (left). (b) A sharp minimum to which a ResNet trained with SGD converged (middle). (c) A wide minimum to which the same ResNet trained with SAM converged(right) (Foret, P. et al. 2020)

It can be seen that by integrating SAM with SGD (Figure (b) and (c)) results in a smoother computational graph which in turn allows for significant error reductions as can be seen in Figure 4(a). In addition to this, SAM also allows for more interpretability in the feature maps of vision transformers as shown by Chen, X., Hsieh, C.-J. and Gong, B. (2021) as well as better reproducability as shown by Somepalli, G. et al. (2022).

Without going into a significant amount of mathematics, the theoretical idea and guarantee in minimising the loss is discussed. By minimising empirical loss, a bound on the true loss arises. The general idea behind SAM is that if the network finds a solution for the weights $\boldsymbol{w}$, then the optimizer will try to perturb this solution by $\boldsymbol{\epsilon}$ where its value lies inside a sphere of radius ρ. Within this sphere, the optimiser finds a point that gives the highest loss where if the network is still able to find a lower loss in this adversarial setting between the networks solution, $\mathbf{w}$, and the optimiser's perturbation, $\boldsymbol{\epsilon}$, then it can be guaranteed that there will be a small value for the expected loss. This is summarised in Equation 9.

$$L_D(\mathbf{w}) \leq \max_{||\boldsymbol{\epsilon}||_2 \leq \rho} L_S(\boldsymbol{w} + \boldsymbol{\epsilon}) + h(\frac{||\boldsymbol{w}||_2^2}{\rho^2}) \tag{9}$$

Where $L_D(\mathbf{w})$ is the true loss, $L_S(\boldsymbol{w}+\boldsymbol{\epsilon})$ is the empirical loss and $h : \mathbb{R}^+ \to \mathbb{R}^+$

is a strictly increasing function under some technical conditions given by $L_D(\boldsymbol{w})$.

The authors state that $h(\frac{||\boldsymbol{w}||_2^2}{\rho^2})$ is computationally expensive and are able to approximate it with a standard L2 regularisation term. Therefore, the simplified bound becomes,

$$\min_{\mathbf{w}} L_S^{SAM}(\mathbf{w}) + \lambda ||\mathbf{w}||_2^2$$

Where,

$$L_s^{SAM}(\mathbf{w}) \triangleq \max_{||\boldsymbol{\epsilon}||_p \leq \rho} L_S(\boldsymbol{w} + \boldsymbol{\epsilon})$$

Eventually, the authors of the paper arrive at the definition of the SAM gradient,

$$\nabla_{\mathbf{w}} L_S^{SAM}(\mathbf{w}) = \nabla_{\mathbf{w}} L_S(\mathbf{w})|_{\mathbf{w}+\hat{\boldsymbol{\epsilon}}(\mathbf{w})} + \frac{d\hat{\boldsymbol{\epsilon}}(\mathbf{w})}{d\mathbf{w}} \nabla_{\mathbf{w}} L_S(\mathbf{w})|_{\mathbf{w}+\hat{\boldsymbol{\epsilon}}(\mathbf{w})}$$

The authors of the paper also state that from empirical results, the derivative can be discarded. The full algorithm that is derived from this is shown in Algorithm 1.

Algorithm 1: SAM algorithm

Input: Training set $\mathcal{S} \triangleq \cup_{i=1}^{n} \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$, Loss function $l : \mathcal{W} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_+$, Batch size b, Step size $\eta > 0$, Neighborhood size $\rho > 0$.
Output: Model trained with SAM
Initialize weights $\boldsymbol{w}_0$, $t = 0$;
while *not converged* **do**
  Sample batch $\mathcal{B} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), ...(\boldsymbol{x}_b, \boldsymbol{y}_b)\}$;
  $\boldsymbol{\delta}(\boldsymbol{w}) = \nabla_{\boldsymbol{w}} L_{\mathcal{B}}(\boldsymbol{w})$;
  $\hat{\epsilon} = \frac{\boldsymbol{\delta}(\boldsymbol{w}_t)}{\|\boldsymbol{\delta}(\boldsymbol{w}_t)\|}$;
  $\boldsymbol{w}_{\text{adv}} = \boldsymbol{w}_t + \hat{\epsilon}$;
  $\boldsymbol{g} = \boldsymbol{\delta}(\boldsymbol{w}_{\text{adv}})$;
  $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \boldsymbol{g}$;
  $t = t + 1$;
end
return $\boldsymbol{w}_t$

3 Proposed Architectures

Given the preliminary knowledge mentioned in Section 2, a novel architecture can be created by combining these concepts. The aim is to create a model that forms better latent representations in comparison to traditional Gaussian VAEs.

3.1 β-SWWDirVAE

The main contibution in this work is to introduce the β Stacked What Where Dirichlet Variational Autoencoder (β-SWWDirVAE) where the β and Dirichlet VAE help to exploit a better latent space and the SWWAE architecture is used to exploit a more stable and reliable autoencoder architecture. The simple representation of the architecture is given in Figure 5.

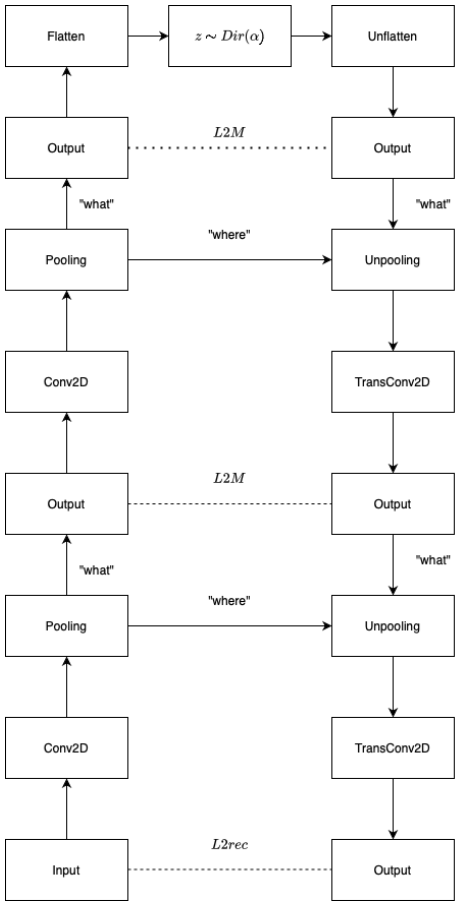


Figure 5: β-SWWDirVAE simplified architecture

As can be seen the core of the architecture is based on the SWWAE model which is modified using DirVAE to make the network act in a more probabilistic manner i.e capture the general distribution of the dataset. In addition to this, the network utilises concepts from the β-VAE which helps to furthr exploit the latent space for a better representation. The loss function for this network is given in Equation 10.

$$\mathcal{L}(\theta, \phi; \mathbf{x}, \mathbf{z}, \beta) = (\lambda_{L2rec} L_{L2rec} + \lambda_{L2M} L_{L2M}) - \beta D_{KL}(q_\theta(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) \quad (10)$$

Where $L_{L2rec} = -\mathbb{E}_{q_\phi(x|z)}[logp_\theta(\mathbf{x}|\mathbf{z})]$ and $D_{KL}(q_\theta(\mathbf{z}|\mathbf{x})||p_\theta(z)) = \sum log\Gamma(\alpha_k) - \sum log(\hat{\alpha}_k) + \sum(\hat{\alpha}_k - \alpha_k)\psi(\alpha_k)$. It should be noted that usually in practice the root mean square error is used instead of the negative log-likelihood.

3.2 β-SWWDirVAE + EffecientNet

Since the previous architecture focuses on creating a better latent representation which contains bettern information than the raw input, this can help supervised learning state-of-the-art architectures to perform better. Past literature shows that for image classification, the EffiecientNet (Tan, M. and Le, Q.V., 2019) has superior performance in comparison to other architectures like ResNet. EfficientNet is a convolutional neural network architecture combined with a scaling method known as "compound scaling".The key idea behind compound scaling is to scale up the three crucial dimensions of a neural network—namely width, depth, and resolution. EfficientNet applies a compound coefficient to scale all these dimensions in a balanced manner, which is what sets it apart from other scaling methods that typically scale only one dimension at a time.

The idea behind adding a β-SWWDirVAE to an EffecientNet is that latent representation from the β-SWWDirVAE will make it easier for the EfficientNet to identify features in images for classification. Such a model would be trained in two steps. First, an unsupervised learning process is performed on the β-SWWDirVAE using large image datasets such as ImageNet. Secondly, the decoder is removed from the β-SWWDirVAE and its weights frozen whereby the encoder of this network is attached to the EfficientNet and trained in a supervised learning manner on CIFAR-10 and Cifar-100 datasets and optimised with SAM. It should also be noted that since the latent space is a vector it would need to be reshaped in order to be compatible with EfficientNet. A simple architecture of this is given below in Figure 6.

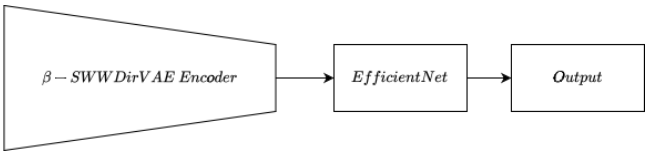


Figure 6: β-SWWDirVAE + EfficientNet simplified architecture

3.3 Additional techniques

In addition to the aformentioned techniques and architectures additional general techniques will be implemented such as augmenting the datasets through rota-

tions, noise, and random cropping. In addition, the networks will also include techniques such as dropout and batch normalisation.

4 Computational Resources

Given the complexity of the models and large amounts of data involved, investment in the appropriate hardware is required. The authors will therefore leverage cloud computing resources such as the one showed here where a large selection of GPUs are available including high performance GPUs such as the H100 and A100.

5 Data Issues

The authors would like to follow in the footsteps of the training procedure of Zhao, J. et al. (2015) where they initially trained the model in an unsupervised manner using the 80 million tiny images dataset and then applied the model to the CIFAR datasets. Unfortunately, the dataset was taken down due to ethical issues regarding bias in the dataset therefore, the authors have proposed two alternatives for the unsupervised learning procedure: a) a small subset of the ImageNet dataset is used (Kaggle) or b) the full ImageNet dataset is used however, this will require a request to access the database. After which, the model can then be further trained in a supervised manner on the CIFAR datasets.

6 Conclusion

In conclusion, several concepts from the field of autoencoders were discussed where the authors believe that the proposed architecture derived from these preliminaries coupled with an existing state of the art optimisation algorithm can improve on existing benchmarks for CIFAR-10 and CIFAR-100 datasets.

7 References

Zhao, J. et al. (2015) "Stacked What-Where Auto-encoders." arXiv. Available at: https://doi.org/10.48550/ARXIV.1506.02351.

Joo, W. et al. (2019) "Dirichlet Variational Autoencoder." arXiv. Available at: https://doi.org/10.48550/ARXIV.1901.02739.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., Lerchner, A. (2017). beta-VAE: Learning Basic Visual Concepts

with a Constrained Variational Framework. International Conference on Learning Representations. https://openreview.net/forum?id=Sy2fzU9gl

Goroshin, R., Mathieu, M., LeCun, Y. (2015). Learning to Linearize Under Uncertainty. CoRR, abs/1506.03011. http://arxiv.org/abs/1506.03011

Foret, P. et al. (2020) "Sharpness-Aware Minimization for Efficiently Improving Generalization." arXiv. Available at: https://doi.org/10.48550/ARXIV.2010.01412.

Tan, M. and Le, Q.V. (2019) "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," arXiv [Preprint]. Available at: https://doi.org/10.48550/ARXIV.1905.11946.

Chen, X., Hsieh, C.-J. and Gong, B. (2021) "When Vision Transformers Outperform ResNets without Pre-training or Strong Data Augmentations." arXiv. Available at: https://doi.org/10.48550/ARXIV.2106.01548.

Somepalli, G. et al. (2022) "Can Neural Nets Learn the Same Model Twice? Investigating Reproducibility and Double Descent from the Decision Boundary Perspective." arXiv. Available at: https://doi.org/10.48550/ARXIV.2203.08124.

Mobahi, H. (2022). 'Sharpness Aware Minimization (SAM) Current State & Future Directions' [PowerPoint presentation] Available at: https://www.dropbox.com/s/66wet9ps2a6i5ey/Hossein_Mobahi_SAM_CSML_Talk.pdf?dl=0