IT UNIVERSITY OF COPENHAGEN

Very Deep Conditional Variational Autoencoders

STADS code: KIP07501PE

August Wester
IT University of Copenhagen
auwe@itu.dk

May 2021

Acknowledgements

This project was made in collaboration with Rasmus Elias Thisted, an MSc student at the University of Copenhagen. All contributions to the project are the result of group effort, with no person deserving more or less credit than the other. This report, however, was written individually. We have neither seen nor discussed each other's written work, which is why this report bears my name alone. In the following, when referring to ”we,” ”our,” ”ours,” or ”us,” I am referring solely to the collaborative effort of the project itself; not the report.

1 Introduction

In the past decade, an astounding amount of progress has been made in the field of deep learning. This has given rise to renewed attention in a diverse range of areas such as computer vision, natural language processing, and generative modelling; the latter of which is the main subject of this report. Concretely, we explore how recent advances in the field of variational autoencoders can be extended to gain more fine-grained control over their generated samples. As we will see, this effort was moderately successful but ultimately ended up posing a greater challenge than we initially expected. We hope this work can potentially serve as inspiration for future work in the area as well as to help others avoid the kinds of pitfalls we encountered along the way.

In the following, we discuss our initial motivation behind the project. We provide provide background on the goals of generative modelling in general as well as the variational autoencoder model in particular. We also introduce recent research into improving variational autoencoders, which has laid the foundation for our project. Finally, we discuss the ideas and experiments we have conducted to extend this research, and we offer a conclusion based on our findings.

2 Motivation

Our work was originally motivated by an effort to improve the robustness of deep learning based image classifiers. In 2014, Szegedy et al. showed how tiny perturbations to the pixel intensity of regular images could cause image classifiers to make catastrophically wrong predictions [1]. For instance, near imperceptible perturbations to an otherwise easily recognizable image of a bird could cause a target model to confidently misclassify it as a truck. These perturbations, known as *adversarial attacks*, were not random but carefully engineered to maximize the loss of the classifier in question. Work on adversarial attacks as well as ways in which models can be protected against them to achieve *adversarial robustness* has since become an active area of research, and much work has yet to be done in this area.

One approach to adversarial robustness is to use so-called preprocessing techniques. These usually rely on a generative model to ”purify” adversarial examples by removing the perturbations before feeding the ”clean” image to a classifier. One example of a preprocessing technique is PuVAE [2] which relies on a so-called *conditional variational autoencoder* (see section 3.3) to reconstruct ”clean” versions of its input. While this works relatively well for simple datasets like MNIST and Fashion-MNIST, robustness rapidly deteriorates as the complexity of the data increases. In the case of PuVAE, we believe this deterioration is in large part due to the model's inability to capture the distribution of more complex datasets like CIFAR-10. (This was, in part, the subject of our bachelor's thesis, written in the spring of 2020.)

Meanwhile, much work has been done in the field of generative modelling, and in variational autoencoders [3] in particular. For instance, Sønderby et. al. and Maaløe et al. achieved impressive results by modifying the architecture of variational autoencoders to learn a hierarchical representation of the data distribution [4, 5]. Most recently, Child explored how that same architecture could be scaled up to 70+ layers (to our knowledge the deepest variational autoencoders explored to date) with even more impressive results, particularly on high-resolution images [6].

We were particularly interested in exploring if and how the PuVAE architecture could be extended to make use of these recent advances. This gave rise to two natural questions:

1. Is it possible to modify the recently proposed variational autoencoder architecture to become conditional?
2. If so, does it lead to increased robustness in the context of more complex datasets?

We believe these are interesting research questions not only in the context of adversarial robustness but in and of themselves as a study in generative modelling. Unfortunately, since addressing the first question posed a greater challenge than we initially expected, the second question remains largely unanswered. The majority of this report is therefore dedicated to the first question, with section 5.3 and 6 providing finalizing thoughts and ideas pertaining to the question of robustness.

3 Background

3.1 A brief introduction to generative models

Although a variety of approaches exist to generative modelling, they are all fundamentally characterized by an attempt to approximate a target distribution $p(\boldsymbol{x})$ using a learned distribution $q(\boldsymbol{x})$, such that novel data points $\boldsymbol{x} \sim q(\boldsymbol{x})$ can subsequently be sampled. For instance, a generative model might be able to capture the distribution underlying a set of images. This would enable it to generate new images that were not part of the original training set but *look* as if they were.

Generative models can be roughly categorized as belonging to either the family of *explicit* or *implicit* models. Explicit models approximate $p(\boldsymbol{x})$ by minimizing an analytically derived loss function, while implicit models estimate $p(\boldsymbol{x})$ without ever explicitly defining it.

One example of an implicit generative model is the generative adversarial network (GAN) [7]. GANs consist of two neural networks: the generator and the discriminator. The purpose of the discriminator is to classify an input image as *real* or *fake* while the purpose of the generator is to generate new, fake images that fool the discriminator into classifying them as real. By optimizing the generator and discriminator in tandem, the authors showed how, in an ideal scenario, the model would reach a Nash equilibrium in which the generator would be so good at generating realistic looking images that the optimal strategy for the discriminator would be to "flip a coin" to determine whether inputs should be classified as real or fake. GANs have since shown impressive results in a wide range of areas; perhaps most notably in the generation of high-fidelity artificial human faces [1]. We explore an example of an explicit model in the next section.

The applications of powerful generative models are vast, and their potential has yet to be fully realized in industry. One could imagine, for instance, how a generative model capturing the distribution underlying a diverse set of human faces could be used to generate a virtually infinite range of realistic-looking characters in a video game. As mentioned, generative models have also shown to be useful in the field of adversarial robustness. Here, models like DefenseGAN [8] and PuVAE [2] have been used to preprocess inputs that may contain tiny adversarial perturbations designed to fool a model, typically by causing a wrong prediction in a classification context. We will return to this topic later.

3.2 Variational autoencoders

A variational autoencoder (VAE) [3] is an explicit generative model. It belongs to the family of *latent variable models* which are characterized by assuming the existence of a hidden or ”latent” random variable $\boldsymbol{z}$ on which the observed variable $\boldsymbol{x}$ is dependent. In the case of VAEs, $p(\boldsymbol{z})$ is usually defined to be a multivariate unit Gaussian distribution $\mathcal{N}(0, I)$ and is referred to as the *prior*. This assumption gives rise to two further distributions: $p(\boldsymbol{x}|\boldsymbol{z})$ (the likelihood) and $p(\boldsymbol{z}|\boldsymbol{x})$ (the posterior). Although neither of these have closed-form expressions which one can optimize, VAEs nonetheless manage to approximate them by formulating the distributions $q(\boldsymbol{x}|\boldsymbol{z})$ and $q(\boldsymbol{z}|\boldsymbol{x})$, the latter of which is referred to as the *approximate posterior*.

It is worth pausing and considering what this formulation entails. Given a sample $\boldsymbol{x} \sim p(\boldsymbol{x})$ as well as good approximate distributions $q(\boldsymbol{z}|\boldsymbol{x})$ and $q(\boldsymbol{x}|\boldsymbol{z})$, we can imagine a concatenated model which infers a latent vector $\boldsymbol{z}$ using the approximate posterior $q(\boldsymbol{z}|\boldsymbol{x})$ and then computes a *reconstruction* $\tilde{\boldsymbol{x}}$ using the approximate likelihood distribution $q(\boldsymbol{x}|\boldsymbol{z})$. Now let $\boldsymbol{z}_{dim}$ and $\boldsymbol{x}_{dim}$ denote the dimensionality of $\boldsymbol{z}$ and $\boldsymbol{x}$, respectively. Then, if $\boldsymbol{z}_{dim} < \boldsymbol{x}_{dim}$, this model will act as a lossy, stochastic compression algorithm for samples from $p(\boldsymbol{x})$, with $q(\boldsymbol{z}|\boldsymbol{x})$ acting as the *encoder* and $q(\boldsymbol{x}|\boldsymbol{z})$ acting as the *decoder*. The generative property of the model becomes apparent when we notice that we can sample latent vectors $\boldsymbol{z} \sim \mathcal{N}(0, I)$ directly and use the decoder to obtain new samples $\boldsymbol{x} \sim q(\boldsymbol{x}|\boldsymbol{z})$ that approximate samples from the true distribution $p(\boldsymbol{x})$.

In practice, the encoder and decoder are usually implemented using a combination of fully-connected and convolutional neural networks. These are trained using backpropagation by minimizing the so-called ”evidence lower bound” (ELBO):

$$\log p(\boldsymbol{x}) \geq \mathbb{E}_{\boldsymbol{z} \sim q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{z})] - D_{KL}[q(\boldsymbol{z}|\boldsymbol{x}) \mid\mid p(\boldsymbol{z})] \tag{1}$$

Here, the first term on the right-hand side corresponds to a reconstruction loss while the second term refers to the KL divergence between the approximate posterior and the prior. Both of these terms have closed form expressions; the first one because we can easily compute a reconstruction loss between the input and the model's output, and the second one because the initial definition $p(\boldsymbol{z}) = \mathcal{N}(0, I)$ allows the KL divergence to be easily computed. For a more in-depth introduction to the ELBO, we refer the reader to the original paper [3].

[1] https://thispersondoesnotexist.com

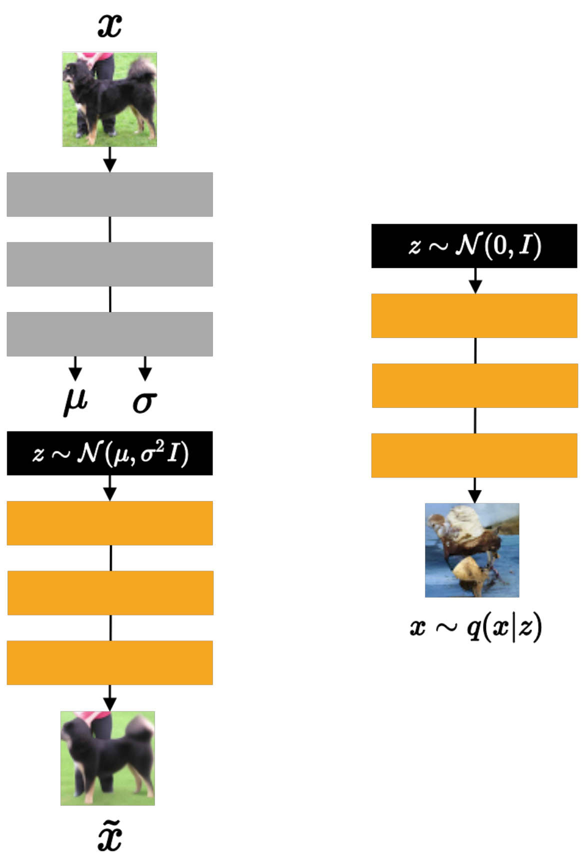
$\boldsymbol{x}$
$\boldsymbol{\mu}$
$\boldsymbol{\sigma}$
$z \sim \mathcal{N}(\mu, \sigma^2 I)$
$\tilde{\boldsymbol{x}}$
$z \sim \mathcal{N}(0, I)$
$\boldsymbol{x} \sim q(\boldsymbol{x}|\boldsymbol{z})$

Figure 1: **Illustration of the difference between training a VAE and using it for image generation.** Left: During training, a VAE learns to minimize the reconstruction loss between its input and output, and the approximate posterior $q(\boldsymbol{z}|\boldsymbol{x})$ is used to sample the latent vector $\boldsymbol{z}$ after running the input through the encoder (gray). The KL divergence term in the model's loss function ensures that $q(\boldsymbol{z}|\boldsymbol{x})$ remains close to $\mathcal{N}(0, I)$. Right: To generate new samples, one simply samples $\boldsymbol{z} \sim \mathcal{N}(0, I)$ and runs it through the decoder (yellow). Assuming the model has been properly trained, the output will be similar to an image from the target distribution $p(\boldsymbol{x})$. Gray and yellow blocks represent neural networks (either fully-connected or convolutional).

Concretely, the encoder outputs two values: a vector $\boldsymbol{\mu}$ of dimension $\boldsymbol{z}_{dim}$ and a scalar $\boldsymbol{\sigma}$. These are used to sample the latent vector $\boldsymbol{z}$ from the distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2 I)$. The KL term in (1) acts as a regularizer by ensuring that $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ don't stray far from $\mathbf{0}$ and 1, respectively. The VAE architecture during both training and generation is illustrated in Figure 1.

3.3 Conditional variational autoencoders

A problem one will encounter using VAEs is the inability to control the characteristics of the generated samples. For instance, in the simple case of modelling the distribution of handwritten digits, one might in some cases be interested in generating only a particular digit. The previous formulation offers no such ability; it merely promises to generate samples that look as if they came from the original distribution given a latent vector $\boldsymbol{z}$, with no added ability to control the finer characteristics of the samples.

This is the problem which the conditional variational autoencoder (CVAE) attempts to solve [9]. The formulation is a simple extension of the regular VAE: Instead of approximating the distributions $p(\boldsymbol{z}|\boldsymbol{x})$ and $p(\boldsymbol{x}|\boldsymbol{z})$, one approximates $p(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{c})$ and $p(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{c})$, where $\boldsymbol{c}$ is a vector that captures some characteristic of the input $\boldsymbol{x}$, usually its ground truth label.

For instance, imagine we want to model the distribution of handwritten digits from the MNIST dataset, and suppose we also want the added ability of controlling which digit to sample. In order to achieve this functionality, we will train a CVAE. Let $\boldsymbol{X} = \{\boldsymbol{x}_0, \boldsymbol{x}_1, ..., \boldsymbol{x}_N\}$ be the training set of handwritten digits and let $\boldsymbol{Y} = \{\boldsymbol{y}_0, \boldsymbol{y}_1, ..., \boldsymbol{y}_N\}$ be the corresponding set of one-hot encoded labels. Finally, let $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ denote the encoder and decoder, respectively. Then, in each iteration of training, instead of running the model $\boldsymbol{\theta}(\boldsymbol{\phi}(\boldsymbol{x}_i))$, we run $\boldsymbol{\theta}(\boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{y}_i), \boldsymbol{y}_i)$. When generating new images belonging to a particular class $\boldsymbol{y}$, we simply ignore the encoder by sampling $\boldsymbol{z} \sim \mathcal{N}(0, I)$ directly and run $\theta(\boldsymbol{z}, \boldsymbol{y})$.

4 The very deep variational autoencoder

As the field of generative modelling has set it sights on higher resolution data in recent years, interest in deeper and more expressive generative models has increased. Unsurprisingly, this interest has given rise to new VAE architectures that are better equipped to capture more complex distributions than what the original formulation is capable of. This development was largely sparked by a common critique of the regular VAE architecture, namely that samples tended to be blurrier than ones from competing generative models such as GANs or autoregressive models [10]. This problem was addressed by Sønderby et. al. when they proposed the ”ladder variational autoencoder” (LVAE) [4]. This architecture consists of a deterministic *bottom-up* path that extracts features from the input and a stochastic *top-down* path in which *multiple* latent vectors $\boldsymbol{z}_0, \boldsymbol{z}_1, ..., \boldsymbol{z}_N$ are sampled at increasing resolutions. Each z_i is made dependent on all $\boldsymbol{z}_{<i}$s that came before it, such that we can express the distribution of the combined latent vector $\boldsymbol{z}$ as

$$p(\boldsymbol{z}) = p(\boldsymbol{z}_0)p(\boldsymbol{z}_1|\boldsymbol{z}_0)p(\boldsymbol{z}_2|\boldsymbol{z}_1, \boldsymbol{z}_0)...p(\boldsymbol{z}_N|\boldsymbol{z}_{<N}).$$

Furthermore, ”lateral” connections between the bottom-up and top-down paths provide the approximate posteriors with access to rich information about the features of the input. Interestingly, the authors showed how the early, low-resolution $\boldsymbol{z}_i$s learn to capture crude, global details while

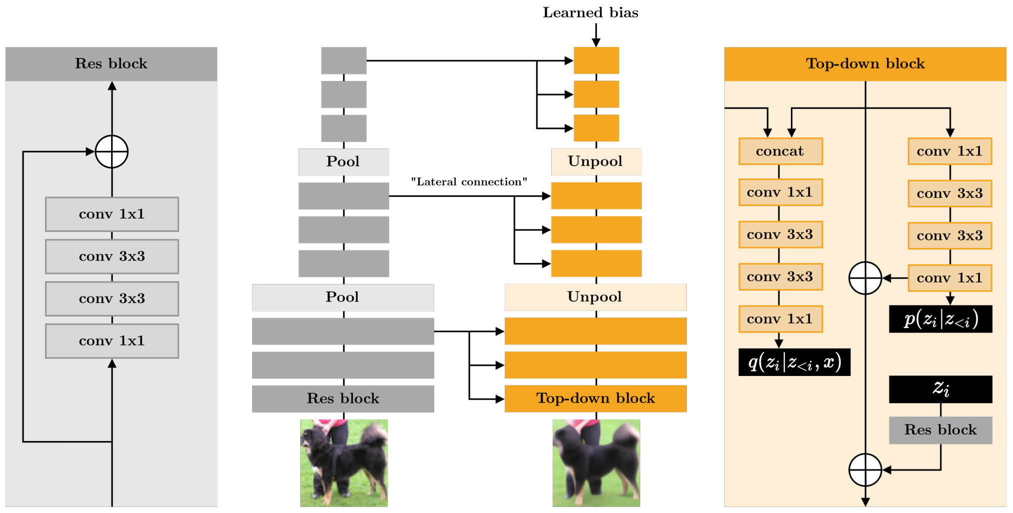

Figure 2: **Architecture of the very deep variational autoencoder (VDVAE)**: The bottom-up path is shown to the left in gray and the top-down path is shown to the right in yellow. $\oplus$ represents addition. This is characteristic of the ”skip connections” in residual networks which allows the number of layers to increase beyond what would otherwise be feasible.

the higher resolution z_is are responsible for filling in finer-grained details in the later stages of the forward pass. This architecture was subsequently extended to 15 layers by Maaløe et. al. [5], the deepest for a VAE at the time, and with promising results that gave renewed attention to VAEs.

Most recently, Child scaled the same architecture to 70+ layers, yielding the ”very deep variational autoencoder” (VDVAE) and showed the most impressive results by a VAE to date [6]. A notable aspect of the architecture is the fact that it uses a *learned* Gaussian prior as opposed to fixing it to a unit Gaussian as discussed earlier. Another perhaps more interesting aspect of the model is the generation of the images themselves. Whereas most VAE decoders perform deconvolutions on the latent vector z in order to transform it into an image of the desired size, the output of the VDVAE is a mixture of logistic distributions *for each pixel in the output image*. In other words, the VAE itself is trained not to reproduce the input image directly but to output a set of pixel-wise distributions that maximize the likelihood of generating the input image. Although this is an interesting departure from the usual workings of VAEs, this is not an aspect of the model we will explore in further detail.

Finally, it is important to note that the VDVAE architecture is fully convolutional. This means that the inputs (in our case images of size $32 \times 32 \times 3$) are never flattened but maintain their 3D structure throughout the model. Note that this does *not* mean that the inputs maintain their size; instead they are processed into feature maps of gradually decreasing resolution in the bottom-up path but with 384 channels. The architecture of the VDVAE is illustrated in Figure 2.

In this work, we have used the VDVAE architecture as our foundation. This is primarily due to two important facts: 1) The model demonstrates the most impressive generative performance for a VAE to date, and 2) the authors have generously shared the associated code, which made it much easier for us to to extend the model and experiment with novel approaches to making it conditional in the sense described in section 3.3.

5 Experiments

5.1 The naive approach: fully-connected layers

As described in section 3.3, the change in formulation between a VAE and a CVAE is small. The difference merely consists in whether or not we include the image's ground truth label as part of the input to the encoder and decoder. However, whereas most VAEs include fully-connected layers which allow for easily incorporating vectorized input (such as a one-hot label) into the flow of the model, the VDVAE is comprised entirely of convolutional layers and thus operates exclusively on 3D volumes. This posed the first challenge, as flat one-hot vectors are incompatible with the model architecture.

A naive approach to this problem would be to simply attach a fully-connected layer in the requisite locations. The volumes could then be flattened, concatenated with the one-hot label, run through a dense layer or two, after which one could reshape the output into the original volume shape and proceed as usual. Unfortunately, this approach quickly becomes intractable. In the late stages of the top-down path, the volumes are of size $32 \times 32 \times 384 = 393,216$, meaning that in the case of 10 classes, concatenating with the one-hot label would yield a vector of size $393,226$. A *single* fully-connected layer would then have to have more than 150 *billion* parameters; more than 3000 times the number of parameters in the whole of the original model (assuming the CIFAR-10 model configuration proposed by the authors). Thus, this approach was quickly disqualified.

5.2 Conditioning using a learned label embedding

A better approach was inspired by work on conditional autoregressive models [11]. Here, the authors make use of a learned label embedding $\boldsymbol{Vc}$ to represent each label using the required dimensionality. (In this case, $\boldsymbol{c}$ represents the one-hot label and $\boldsymbol{V}$ represents the matrix of trainable parameters.) We translated this idea to the VDVAE by replacing the learned bias at the top of the top-down path with an embedding layer taking a one-hot encoded label as input. Since the resolution in the top of the top-down path is $1 \times 1 \times 384$, computing a flat 384-dimensional embedding and reshaping it to $1 \times 1 \times 384$ was sufficient to make this work. This turned out to be the first, somewhat successful, approach to making the VDVAE conditional.

As in a traditional CVAE, we associate each input image with its ground truth label during training. Over time, as the model is trained, the embedding associated with each class is gradually updated so as to better contribute to making the reconstructions closer to the original. Given that each embedding only contributes to reconstructing images belonging to its associated class, it should end up acting as a signal to the model to produce an image of that class once the model has been trained.

To test this approach, we modified the architecture as described, and trained the resulting model on the CIFAR-10 dataset for 350 epochs. This took about two weeks using Google Colab Pro

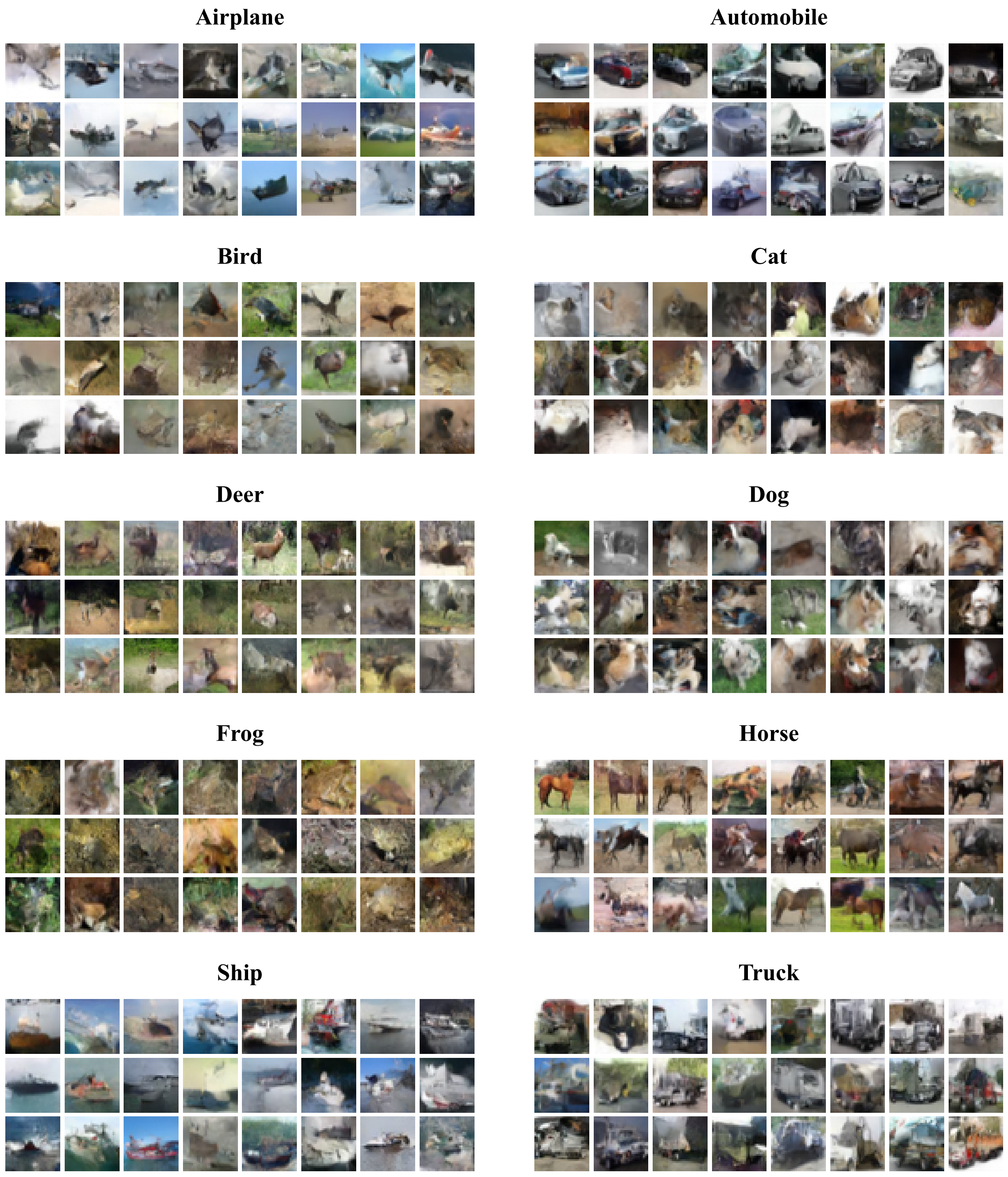

Figure 3: **24 non-cherrypicked, temperature 0.85 samples for each of the ten classes in CIFAR-10.** The samples were generated using the learned priors of the modified VDVAE with a label embedding. We found that samples at temperature 1.0 tended to display excessive variance, while 0.85 made the samples look more realistic.

with priority access to their T4 and P100 GPUs. Once the model had been trained, we sampled 24 images belonging to each of CIFAR-10's classes. The results are shown in Figure 3. It is clear from this that the model has learned to associate each embedding with structures that are common to images belonging to its corresponding class (although some classes are seemingly harder to model than others).

When sampling, we initially found the resulting images to display too much variance and too little global structure, which made the images look less realistic. To address this, we followed the approach in [6] and adjusted the sampling *temperature*. Temperature is specified using a scalar $\boldsymbol{t}$ which influences the sampling operation by reducing the variance of the Gaussian priors:

$$\boldsymbol{z}_i \sim \mathcal{N}(\boldsymbol{\mu}_i, \sigma_i^2 + \log \boldsymbol{t})$$

Thus $\boldsymbol{t} = 1$ is the implicit default whereas $\boldsymbol{t} < 1$ and $\boldsymbol{t} > 1$ will reduce or increase the variance, respectively. We found $\boldsymbol{t} = 0.85$ to be a suitable "sweet spot" between enough variance and enough global structure.

It is interesting to note that although many of the samples in Figure 3 look somewhat realistic, they are not nearly as impressive or high-fidelity as the samples of human faces in the original paper [6]. Clearly, much of this has to do with the fact that we are using a smaller model operating on 32×32 images. However, looking at the paper's ImageNet-32 (a dataset similar to CIFAR-10 but with 1000 classes instead of 10) reconstructions, we see that a majority of these lack global structure and don't clearly belong to one class or another ([6], Figure 10). This raises an interesting question: How come their model can effortlessly generate high-resolution human faces while a corresponding model trained on ImageNet-32 struggles with generating realistic-looking images of considerably lower resolution? We believe this is because datasets like CIFAR-10 and ImageNet-32 are, in some sense, harder datasets to model due to the high variance in image characteristics both within and between classes. This variance does not carry over to human portraits to the same extent. Here, the model can simply learn to generate a pair of eyes in the top-half of the image, a mouth in the bottom half, and so on.

Please note that we have shared much of the code associated with this model as a notebook on Google Colab [2]. This will allow you to run the model yourself and generate an endless stream of new samples. The trained weights used for the samples in Figure 3 will be loaded into the model, so you do not have to train the model yourself.

Now that we have demonstrated how to achieve a conditional version of the VDVAE, we turn our attention to how this could possibly be used to achieve adversarial robustness.

5.3 Partial sampling of conditional priors during reconstruction

One problem with the previous approach is the label's lack of influence on the final image when reconstructing inputs to the model. Usually this would not be seen as an issue, as generative modelling is mostly concerned with generating entirely new samples, as opposed to finding a "compromise" between an input image and a label. (This is why the encoder is usually discarded after training a VAE.) However, in order for the model to be useful as an alternative to

[2] https://colab.research.google.com/drive/1MmwBizAD5eo-5IKNgslE4QlCy6pK7PHW

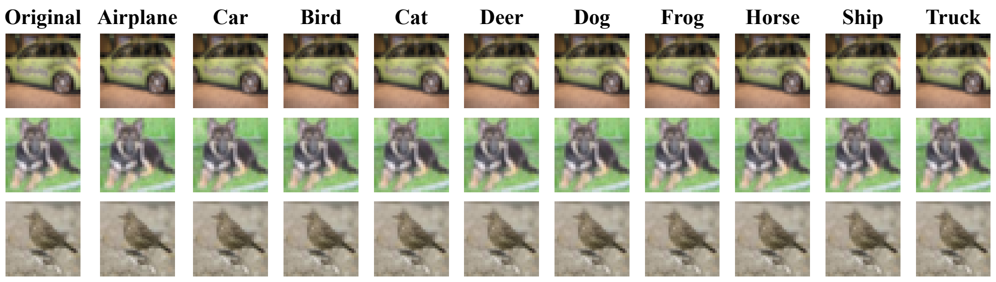

Figure 4: **Reconstructions using 10 different label embeddings.** During reconstruction, when the approximate posterior is used, the embedding has virtually no influence on the final image. Instead, the approximate posterior relies on its input from the bottom-up path which enables near perfect reconstructions.

the ”vanilla” VAE used in the PuVAE architecture, labels will have to exert a greater influence over the model’s reconstructions (not just its purely generative samples). This is because PuVAE works by taking a potentially adversarially perturbed input and reconstructing it once for each possible class (10 in the case of CIFAR-10) after which it picks the reconstruction minimizing the mean squared error with the original image and feeds it to a classifier for classification. We refer the reader to the original paper [2] for more details but stress that the property expected by PuVAE is that 1) the label influences the reconstruction, and 2) ideally in such a way that when an input is paired with its correct label, the reconstruction will be better than when paired with any other label. Realistically, however, what we seek is a model which given an input image $\boldsymbol{x}$ paired with an *arbitrary* label $\boldsymbol{y}$ will produce a reconstruction $\tilde{\boldsymbol{x}}$ which is a compromise between the visual characteristics of $\boldsymbol{x}$ and the semantics of $\boldsymbol{y}$.

In the case of our model, we hypothesize the underlying reason for the label’s lack of influence on the final reconstruction to be the following: Since the approximate posteriors are sampled during reconstruction, and since they have access to rich information about the original image due to the model’s ”lateral” connections, the approximate posteriors already have enough information about the input to be able reconstruct it more or less perfectly (we have verified this empirically, which is shown in Figure 4). An added label embedding will thus act more as a source of noise which the model eventually learns to ignore during reconstruction.

We are thus faced with an apparent trade-off: Either 1) restrict the amount of information transferred between the bottom-up and top-down paths and weaken the generative power of the model with the added benefit of labels exerting greater influence over the outputs, or 2) keep the lateral connections intact at the cost of labels being ignored during reconstruction.

Neither of these seem like attractive options. However, the problem gave rise to the following idea for an alternative approach making use of the hierarchical structure of the VDVAE: Rather than having a single label embedding at the top of the top-down path, introduce one label embedding *per top-down block* and add this as a bias to the input to both the approximate posterior and the prior. Once the model has been trained, denote the number of top-down blocks by N. During reconstruction, sample from the first $k < N$ approximate posteriors, and for the remain-

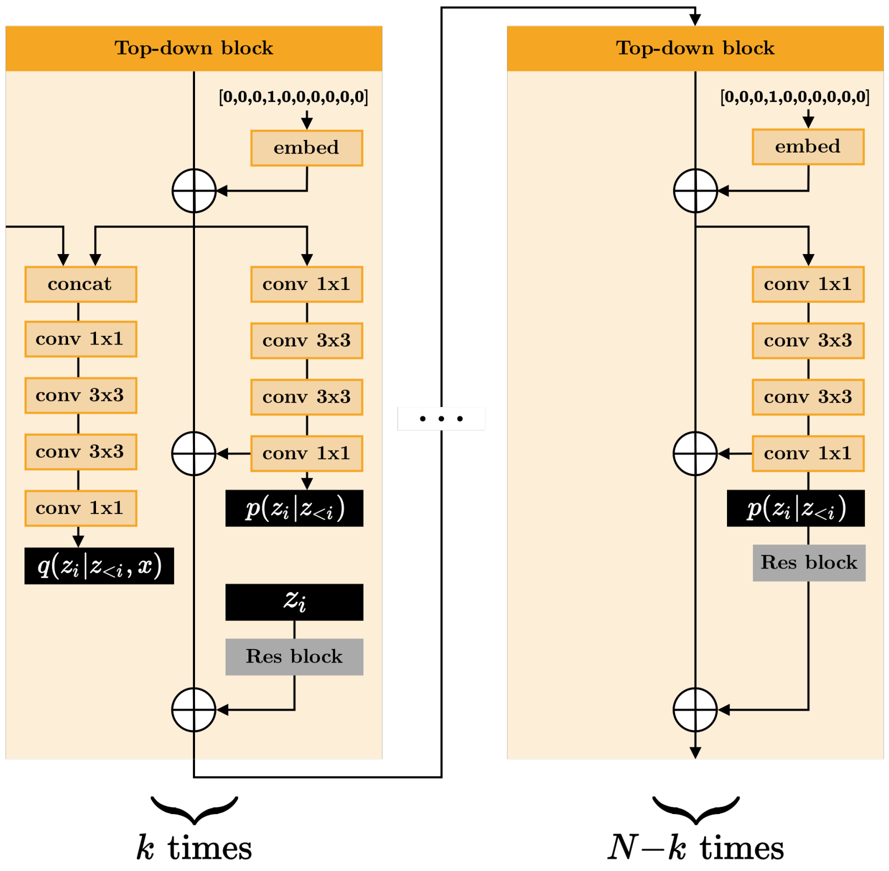

Figure 5: **Proposed method for partially sampling from conditional priors during reconstruction:** By restricting sampling to the first k approximate posteriors, with the remaining $N-k$ samples being solely from conditional priors, we hoped the model would preserve true global image characteristics while also filling in label-specific details.

ing forward pass, sample from the following $N - k$ priors until the image is reconstructed. This construction is visualized in Figure 5.

Why should this work? As illustrated in Figure 2, the lateral connections from the bottom-up path to the top-down path do not transfer to the prior. Thus, when k is sufficiently low, the priors will have to ”fill in” the remaining details on their own, without access to information about the input from the bottom-up path. Assuming the label has influence over the data generated by the prior (which Figure 3 strongly suggests is the case), sampling from the first few approximate posteriors will produce $z_{i<=k}$ that capture many of the input's true global features while the remaining priors will have to fill in more fine-grained *but label-specific* details, unconditional on all but the most crude features of the input image.

To test this approach, we trained the new model in much the same way as the one proposed in the previous section. However, this time it took almost twice as long (more than three weeks) using Google Colab Pro. This was due to the increased number of parameters introduced by the new embedding layers. (In hindsight, this could probably have been implemented in a more economical way, such as by reducing the size of the embeddings and scaling them up to the required size using deconvolution. This would have reduced the number of parameters and likely decreased the training time to a little less than two weeks.)

After the model had been trained, we drew 16 random samples from the test set and reconstructed each 10 times using different labels with $k = 2$. Unfortunately, we did not obtain the results we were hoping. Although the reconstructions displayed inter-class variance (as opposed to the ones in Figure 4), the majority lacked discernible global structure, and the ones corresponding to the input's actual class label did rarely resemble the original. The results are shown in Figure 6.

We suspect the reason for the poor performance partly lies in the fact that each embedding layer is only conditioned on the one-hot label and not on any of the latent vectors sampled in the upper layers of the top-down path. This makes each embedding deterministic and fixed once the model has been trained. This is likely too rigid of a structure for a model so reliant on sampling. Had we had more time, we would have probably experimented with ways of conditioning each label embedding on the preceding ones. This would also be in line with the hierarchical structure of the model as a whole.

With that said, it is important to emphasize that this particular problem—transforming an image of one class into an image of another while preserving global characteristics—is an extremely hard problem, and is not something for which we are familiar with related research.

6 Conclusion

In this project, we have explored the problem of making the very deep variational autoencoder conditional. In its original formulation, the generative capabilities of this model are limited to unconditional samples; that is, samples over which one does not have control over the image characteristics. We proposed an approach to making the model conditional by replacing the model's learned bias by an embedding layer taking a one-hot encoded label as input. We found that after training this model on the CIFAR-10 dataset, the learned embeddings were successful in having

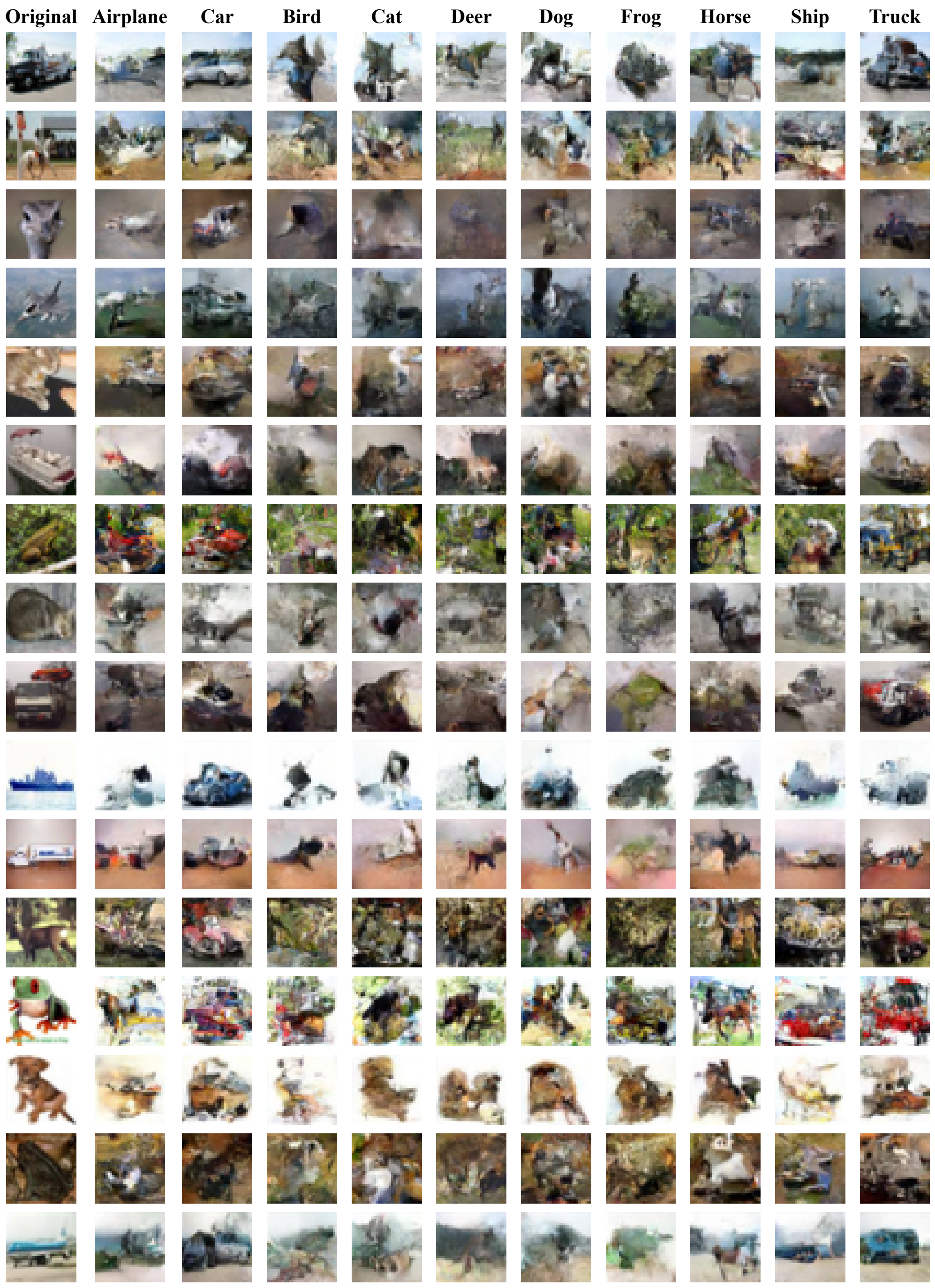

Figure 6: **Reconstructions of 16 images from the test using** $k = 2$ **and** $t = 0.85$**.**

the label control the semantics of the outputted images, although with somewhat differing fidelity depending on the class.

As mentioned in section 2, our initial motivation for this project was sparked by an interest in improving the adversarial robustness of deep learning based image classifiers beyond what we achieved in our bachelor's thesis. Specifically, we wanted to improve the efficacy of the PuVAE architecture using a larger CVAE with significantly higher generative capacity. What we found was that the key property expected by PuVAE—namely the ability of the label to influence the reconstruction of the input image into an image of the corresponding class—is hard, even for a much bigger model like the VDVAE. Of course, we first had to make the VDVAE conditional, and it is entirely possible that there are better ways to do this than our proposed solution. However, we interpret our results as an indication that the approach to adversarial robustness outlined by PuVAE does not scale to more complex datasets, and should be avoided for any image recognition tasks outside of very simple datasets like MNIST.

Had we had more time, we would have liked to experiment with alternative approaches to the problem of conditional reconstructions, although we have become somewhat skeptical of the prospects of this technique for improving adversarial robustness. We would also have been interested in applying the technique in section 5.2 to even bigger models and more high-resolution datasets. Unfortunately, the size of such a model would have been unfeasible for us, given our limited time and computing resources.

References

[1] Christian Szegedy et al. *Intriguing properties of neural networks*. 2014. arXiv: 1312.6199 [cs.CV].

[2] Uiwon Hwang et al. *PuVAE: A Variational Autoencoder to Purify Adversarial Examples*. 2019. arXiv: 1903.00585 [cs.LG].

[3] Diederik P Kingma and Max Welling. *Auto-Encoding Variational Bayes*. 2014. arXiv: 1312.6114 [stat.ML].

[4] Casper Kaae Sønderby et al. *Ladder Variational Autoencoders*. 2016. arXiv: 1602.02282 [stat.ML].

[5] Lars Maaløe et al. *BIVA: A Very Deep Hierarchy of Latent Variables for Generative Modeling*. 2019. arXiv: 1902.02102 [stat.ML].

[6] Rewon Child. *Very Deep VAEs Generalize Autoregressive Models and Can Outperform Them on Images*. 2021. arXiv: 2011.10650 [cs.LG].

[7] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[8] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. *Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models*. 2018. arXiv: 1805.06605 [cs.CV].

[9] Carl Doersch. *Tutorial on Variational Autoencoders*. 2021. arXiv: 1606.05908 [stat.ML].

[10] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. *Pixel Recurrent Neural Networks*. 2016. arXiv: 1601.06759 [cs.CV].

[11] Aaron van den Oord et al. *Conditional Image Generation with PixelCNN Decoders*. 2016. arXiv: 1606.05328 [cs.CV].