Masterarbeit am Institut für Informatik der Freien Universität Berlin

An Investigation into Image Area Dependencies using Unsupervised Learning

Leon Sixt

leon.sixt@fu-berlin.de
Matrikelnummer: 4551948
Berlin, April 5, 2018
1. Gutachter: Prof. Dr. Tim Landgraf, Freie Universität Berlin
Betreuer & 2. Gutachter: Prof. Dr. Matthias Bethge, Universität Tübingen

Abstract

I present my investigations into explicitly mapping dependencies between image areas and identify limitations of that approach. I continue with an analysis of variational autoencoders. Building on the insights, I train VAEs on different sized image patches extracted from the CIFAR-10 and CelebA datasets. The patch VAEs are used to compute a similarity metric between individual patches. As an evaluation, linear models are trained to predict the labels given the unsupervised representations. The patch-VAEs allow to obtain better classification accuracies then using the representations of a global VAE on CIFAR-10 and CelebA. I find that for VAEs the features of the earlier layers give a better linear classification performance than their representations.

Eidesstattliche Erklärung

Ich versichere hiermit an Eides Statt, dass diese Arbeit von niemand anderem als meiner Person verfasst worden ist. Alle verwendeten Hilfsmittel wie Berichte, Bücher, Internetseiten oder ähnliches sind im Literaturverzeichnis angegeben, Zitate aus fremden Arbeiten sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungskommission vorgelegt und auch nicht veröffentlicht.

April 5, 2018

Leon Sixt

Contents

1 Introduction

Deep learning aims to solve various, complex tasks such as object detection for autonomous driving, medical image analysis, or language translation. One major obstacle employing deep learning in those applications is its hunger for data. Complex dependencies exist between data and labels. Learning the dependencies requires many annotated samples. As humans have to label the data usually, faster adoption would be possible if fewer samples were needed. Here, unsupervised learning enters the stage. Its main promise is to learn a more useful representation of the data without requiring labels. Given a good representation, learning the connection from the representation to the labels should be easier.

Images contain structure at different scales, e.g. from edges over eyes to whole faces. It would be useful to have an unsupervised learning algorithm that can identify structures and their dependencies at different scales. However, popular unsupervised models operate on the whole image which makes it hard to gain knowledge about dependencies within an image. For example, PixelCNN (Oord et al., 2016a) show very impressive log-likelihood scores but their knowledge of the dependencies between areas is hidden within their complex weights.

In this work, I discuss two approaches to uncover local dependency structure. First, I show how dependencies can be measured using the pointwise mutual information. The probability distributions involved in computing the pointwise mutual information are approximated using PixelCNN models. I discuss multiple issues which arose using this approach.

For the second approach, I first analyze VAEs (Kingma and Welling, 2013) from an information theoretic perspective and discuss how coding inefficiencies and the independence structure of the decoder influence the information stored in the latent variables. Building on the insights, I construct a local patch VAEs with various patch sizes and independence structures.

I evaluate the learned representation of the patch VAEs on the CelebA (Liu et al., 2015) and CIFAR-10 (Krizhevsky, 2009) datasets. For both datasets, I train linear classifiers on the learned representations to predict the face attributes for CelebA and the class labels of CIFAR-10. For CelebA, the representation of the patch-VAE allows a significantly better prediction performance over global VAEs. It comes even close to a supervised baseline. I also show that learned representation space contains meaningful visual similarities, e.g. eye patches are similar. For CIFAR-10, the performance of all tested unsupervised models are far from the supervised baseline away. Still, patch-VAEs perform better than the representations of global VAEs.

In the next section 2, I present the relevant related work. I then discuss the explicit learning of area dependencies and its limitations in section 3. In section 4, a detailed analysis of VAEs is presented. Section 6 and 7 show the experiments setup and the results. In the last section 8, I discuss the work and give an outlook over future work.

2 Related Work

In unsupervised deep learning, there are currently three main models: variational autoencoders (VAEs) (Kingma and Welling, 2013), generative adversarial networks (GANs) (Goodfellow et al., 2014), and autoregressive models such as PixelRNN/CNN (Oord et al., 2016b; Oord et al., 2016a).

While current enhancements of GAN such as (Karras et al., 2017) and (Arjovsky et al., 2017) produce very real looking images, a major drawback of GANs is that they do not provide an estimate of the data probabilities. However, as I will formulate area dependencies using probability densities, it is not possible to use GANs for my task.

An advantage of VAEs is that they compress data samples to a latent code. There are different models to adapt the information stored in the latent code. The β-VAE introduces an additional hyperparameter to decrease the stored information. In the variational lossy autoencoder (VLAE) (Chen et al., 2016), the dependency structure is modified to gain finer control over what is represented in the latent code. Another recent line of research in VAEs are hierarchical models (Sønderby et al., 2016; Zhao et al., 2017; Bachman, 2016).

Autoregressive models currently show the best log-likelihood scores on complex image distributions. A starting point for autoregressive models was the work by Theis and Bethge, 2015 who used spatial LSTM to model image texture. Oord et al., 2016b applied autoregressive models to resized images from ImageNet (Deng et al., 2009) and CIFAR10 datasets and extended them by a new RNN architecture. In the PixelCNN paper (Oord et al., 2016a), a CNN is used instead of a recurrent network and a conditional version is proposed. Salimans et al., 2017 propose many useful changes to simplify the structure and improve the performance of PixelCNNs.

A disadvantage of autoregressive models is that they miss a latent representation. All learned dependencies are implicitly encoded into the weights of the neural networks. PixelCNN models achieve stunning log-likelihood scores which means they can explain the data well. However, their understanding is implicitly stored in the complicated weights and hence hidden from inspection.

Direclty mapping image area dependencies is an undeveloped field. Lindgren et al., 2008 studied local luminance and contrast dependencies between patches from natural images. To my best knowledge, explicitly mapping the dependencies between areas was not tried before.

3 Explicit Mapping of Area Dependencies

The mathematical notation for probabilities used throughout this thesis is the following. A random variable X can take values from a set $\mathcal{X}$, and it has a probability distribution $P(X)$. The probability density function is denoted by $p(x)$ where x is a single outcome of X.

At first, I wanted to map the dependencies between areas explicitly. In information the-

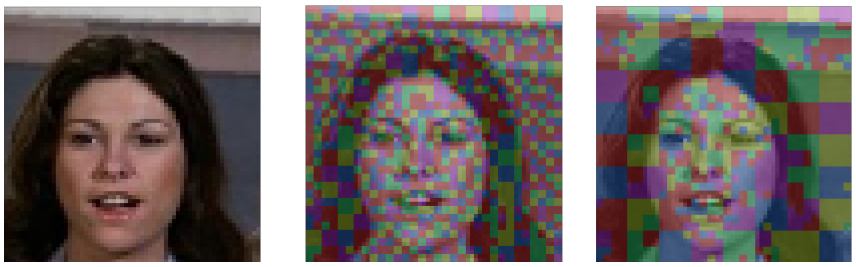

Figure 1: Merging areas based on the pmi did not result in meaningful larger areas. Each merged areas is colored differently.

ory, the mutual information $I[X;Y]$ measures the dependency between random variables. Higher mutual information means higher dependency. As I am interested in the dependency between two areas of an image, I need the pointwise mutual information that measures the dependencies between the outcomes of two random variables:

$$\mathrm{pmi}(x,y) = \log \frac{p(x,y)}{p(x)p(y)} = \log \frac{p(x|y)}{p(x)} \tag{1}$$

For two areas x and y, the pmi quantifies the information gained when one is observed. As the true data generating probability distribution is unknown only a set of samples is known, I approximate $p(x)$ and $p(x|y)$ from image data using PixelCNN models. I call the learned approximation $\hat{p}(x)$ and $\hat{p}(x|y)$. If they converge to the true data generating probability distributions then:

$$\mathrm{pmi}(x,y) \approx -\log \hat{p}(x) + \log \hat{p}(x|y). \tag{2}$$

In theory, PixelCNNs should give good approximations of both probability distribution and therefore also of the approximated pmi score should be close to the true pmi. However, I encountered multiple obstacles with mapping the dependencies explicitly.

As an area can be any subset of the image, there are $2^n - 1$ possible areas where n is the number of pixels. The set of possible combinations is therefore too large to map all dependencies. At a smaller scale, image areas are highly dependent on their neighbors which reduces the areas in question significantly. Therefore, I started at the pixel level and recursively merged neighboring areas if they have a large pmi.

The recursive merging resulted in non-rectangle areas. Thus, a problem was to estimate the probability densities for a non-rectangular subset which is equally to estimate the marginal distribution of a rectangular patch. For example, the probability distribution of a PixelCNN fitted on 2x2 patches would factorize as $p(x_1)p(x_2|x_1)p(x_3|x_2,x_1)p(x_4|x_3,x_2,x_1)$ where x_i is the i-th pixel. Estimating the marginal distribution $p(x_1,x_2)$ would be easy as we just run the chain until $p(x_1)p(x_2|x_1)$. However, the marginal distribution $p(x_2,x_4)$ is hard to obtain. The PixelCNN only learned $p(x_2|x_1)$, $p(x_4|x_3,x_2,x_1)$ and unfortunately x_1, x_3 are not known in this case. A method to still estimate marginals like $p(x_2,x_4)$ would be through sampling x_1 and x_3 – an expensive and imprecise method.

The assumption was that combining areas with large pmi will result in areas sharing a lot

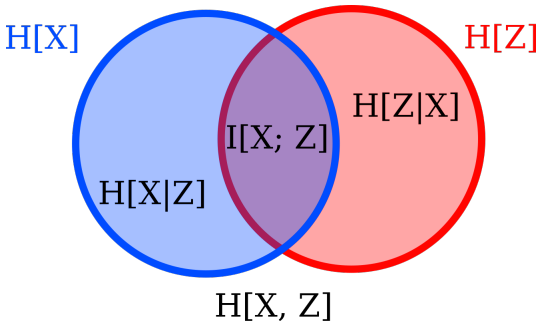

Figure 2: Venn diagram of information theory. The entropy of the individual random variables $H[X], H[Z]$ are the blue and red circles. Their overlapping contains the mutual information $I[X; Z]$. The conditional entropies $H[X|Z], H[Z|X]$ are the areas where they do not overlap. And the joint entropy $H[X, Z]$ is the joint area of the circles.

of information. For example, areas containing the same hair texture being merged together. However, this was not the case. Indeed, I found that patches containing edges share much more information then patches from the same texture which leads to areas of edges being merged together and not similar areas. In Figure 1, an exemplary image with merged areas of are shown.

Another problem was that I considered only the pmi between two areas at a time. Complex structures such as eyes couldn't be found because the individual smaller subareas within an eye do not share much information.

As an alternative, I look at how VAE could be used to gain insights into the local dependency structure. In the next section, I analyze and review VAEs. Especially, I investigate what information will be stored in the latent code of a VAE.

4 Analysis of Variational Autoencoder

In the previous section, I discussed an approach to map image dependencies explicitly. One problem was the merging of areas. It could be useful to have a similarity score to combine similar areas fast together. Here, VAEs could prove useful as the distance of samples in the latent space can be used as a similarity score.

I first review the addition of Gaussian noise as a simple information bottleneck. Then, I show that an autoencoder using an information bottleneck with fixed and too little capacity will encode information that is shared within many data variables.

Building on the insights from the simple case, I continue to discuss which information a VAE encodes into the latent variables. I give an explanation why VAE fails when using autoregressive models as decoders and analyze the β-VAE architectures.

Table 1: Probability distributions that appear in capacity autoencoders and variational autoencoders.

Prob. Dist	Description
$P_D(X)$	the true data distribution (unknown, but samples are given).
$Q(Z\|X)$	the inference distribution (parameterized by a NN).
$P(X\|Z)$	the decoding distribution (parameterized by a NN).
$P(X)$	the model data distribution (unknown but can be upper-bounded for VAEs).
$Q(Z)$	corresponds to $q(z) = \int q(z\|x)p_D(x)dx$ (unknown).
$Q(X\|Z)$	the perfect decoder. Given by $q(x\|z) = \frac{q(z\|x)p_D(x)}{q(z)}$ (unknown).

4.1 Capacity Autoencoder

Here, I will introduce a simple autoencoder that will make it easy to analyze which information is transmitted in the representation.

We have for any random variable Z that:

$$H[X] = H[X|Z] + I[X;Z]. \tag{3}$$

This equation is the starting point of an autoencoder. The random variable Z will be the learned representation of X. I choose $Z = f(X) + \epsilon$ where $\epsilon \sim \mathcal{N}(0, nI)$ is Gaussian noise with the same variance in all dimensions n^2.

The conditional entropy $H[X|Z]$ will be approximated using a decoder. Using the decomposition of the conditional entropy into cross entropy and KL-divergence, $H[X|Z] = \mathbb{E}_{X\sim P_D}[P(X|Z)] - D_{KL}(Q(X|Z)||P(X|Z))$ gives:

$$H[X] + D_{KL}(Q(X|Z)||P(X|Z)) = \mathbb{E}_{X\sim P_D}[P(X|Z)] + I[X;Z] \tag{4}$$

where $P(X|Z)$ is the decoder distribution and $Q(X|Z)$. An overview of the different probability distributions is given in Table 1. In equation 4, the data entropy $H[X]$ is fixed. The LHS KL-term cannot be approximated as $Q(X|Z)$ is unknown. The cross entropy term $\mathbb{E}_{X\sim P_D}[P(X|Z)]$ can be calculated. The decoder distribution is parameterized by a neural network and both Z and X can be sampled, i.e. $Z \sim Q(Z|X)$ and samples of X are given by the train set. I plan to minimize the RHS as this will minimize the RHS KL-divergence.

The mutual information term $I[X;Z]$ is left to approximate. Usually, information theory quantities are hard to estimate. However, there exists a surprisingly simple formula to upper bound the maximal mutual information if also the variance of $f(x)$ is bounded. Therefore, I require that the variance of $f(x)$ is $\text{Var}[f_i(X)] = s^2$ for every dimension. Otherwise, the noise could be ignored by producing large values. Batch normalization can fix the variances in practice.

4.1.1 Maximal Capacity

I have picked $Z = f(X)+\epsilon$, i.e. the addition of a deterministic function and an independent noise vector. What is the maximal amount of information about X that can be transmitted in Z? The quantity of maximal information is called channel capacity C and its mathematical formulation is:

$$C = \sup_{\text{s.t. Var}[f_i(X)]=s^2} I[X;Z]. \tag{5}$$

The encoder should maximize the mutual information between X and Z while having a fixed variance. I begin by rewriting the mutual information as entropies:

$$\begin{aligned} I[X;Z] &= H[Z] - H[Z|X] \\ &= H[Z] - H[f(X) + n\epsilon|X] \\ &= H[Z] - H[n\epsilon] \end{aligned} \tag{6}$$

Here, I used that f is a deterministic function and therefore entirely determined by knowing X, i.e. $H[f(X)|X] = 0$. I know that $\text{Var}[Z_i] = s^2 + n^2$ as Z is the addition of two independent random variables. The mean is arbitrary and can be set to $\mathbb{E}[Z_i] = 0$. For a fixed mean and variance, the maximum entropy distribution is given by a Gaussian distribution.

$$\begin{aligned} &= H[Z] - H[n\epsilon] \\ &\leq H[\mathcal{N}(0, (s^2 + n^2)I^k)] - H[n\epsilon] \end{aligned} \tag{7}$$

The differential entropy of the Gaussian distribution is $H[\mathcal{N}(0, \Sigma)] = \frac{1}{2}\log\left(\det(2\pi e\Sigma)\right)$:

$$\begin{aligned} &= H[\mathcal{N}\left(0, (s^2 + n^2)I^k\right)] - H[n\epsilon] \\ &= \frac{k}{2}\log\left(2\pi e(s^2 + n^2)\right) - \frac{k}{2}\log\left(2\pi e n^2\right) \\ &= \frac{k}{2}\log\left(\frac{s^2 + n^2}{n^2}\right) \\ &= \frac{k}{2}\log\left(1 + \frac{s^2}{n^2}\right) \end{aligned} \tag{8}$$

Thus, we have that the maximal number of bits that Z can contain about X is:

$$C = \frac{k}{2}\log\left(1 + \frac{s^2}{n^2}\right) \tag{9}$$

That is exactly the channel capacity of an additive white Gaussian noise channel and was first derived by Shannon, 1948. The coding inefficiency is the difference between the mutual information $I[X;Z]$ and the channel capacity:

$$C - I[X;Z] = H[\mathcal{N}(0, (s^2 + n^2)I^k)] - H[Z] \tag{10}$$

The optimal encoding would be achieved if the distribution $f(X)$ is Gaussian. However, even for an optimal encoding, the maximum number of bits that are transmitted is restricted by eq. (9).

4.1.2 Information stored in Capacity Autoencoder

When the capacity of the channel is not sufficient to transmit all information, what kind of information will the encoder transmit? Clearly, the encoder has to discard some information but how will this decision be made? And how can this decision be influenced?

The amount of information is set fixed to C but I assume that the encoder has full power over which information to transmit. The most important factor is then the independence structure of the decoder. For the case, that the decoding distribution is fully factorized, i.e. $\log p(X|Z) = \sum_i \log p(X_i|Z)$, it is beneficial to transmit information that is shared between multiple data coordinates. Encoding one bit of information shared with k data coordinates can decrease the decoding loss up to k-bits.

Whereas if the decoding distribution can model any dependencies such as in autoregressive PixelCNN models, the encoder has full freedom over which information to transmit.

If the independence structure of the decoder makes it impossible to model the data distribution fully, the encoder has to fill in the missing information about the dependencies. Therefore, the encoder will provide the information that is most useful for the decoder to match the data distribution. Moreover, using the dependency structure it can be control what kind of information is useful. For example, if using as decoder independence assumption that 3x3 patches are independent, local variations will be fully modeled within the patches and larger dependencies should be represented in Z.

4.2 Variational Autoencoder

In VAEs, the encoder can additionally control the noise level and is given by:

$$Z = f_\mu(X) + f_\sigma(X)\epsilon \tag{11}$$

Compared to capacity autoencoders, this means that the amount of information transmitted is determined by the optimization process.

The main equation of a VAE is:

$$\begin{aligned} \log p_\theta(X) - & D_{KL}[Q(Z|X)||P(Z|X)] = \\ & = \underbrace{\mathbb{E}_{Z\sim Q(Z|X)}[\log p_\theta(X|Z)]}_{\text{reconstruction}} - \underbrace{D_{KL}[Q(Z|X)||P(Z)]}_{\text{regularization}} \end{aligned} \tag{12}$$

where the RHS is maximized. As the $D_{KL}[Q(Z|X)||P(Z|X)]$ is nonnegative, the RHS is a lower-bound of the model log-likelihood:

$$\log p_\theta(X) \geq \mathbb{E}_{Z\sim Q(Z|X)}[\log p_\theta(X|Z)] - D_{KL}[Q(Z|X)||P(Z)] \tag{13}$$

Therefore, maximizing the RHS will also maximize the model likelihood. To better understand VAEs, I will analyze the expectation of the main VAE equation (12):

$$\begin{aligned}\mathbb{E}_X[-\log p_\theta(X)] + \mathbb{E}_X[D_{KL}[Q(Z|X)||P(Z|X)]] = \\ = \mathbb{E}_{X,Z\sim Q(Z|X)}[-\log p_\theta(X|Z)] + \mathbb{E}_X[D_{KL}[Q(Z|X)||P(Z)]]\end{aligned} \tag{14}$$

Here, I also negated to the equation such that it is in a similar form as entropies. The reconstruction term is simple to understand. It measures how well the decoder $p_\theta(X|Z)$ can reconstruct X given a Z sampled from $Q(Z|X)$. However, understanding the meanings of the KL-divergences is more complicated.

In the next section, I rewrite the KL-term on the RHS and relate it to the channel capacity. Then I proceed to analyze the KL-term on the LHS and relate it to the coding inefficiencies of a VAE.

4.2.1 Channel Capacity of a Variation Autoencoder

The right KL-term from eq. (14) can be rewritten as:

$$\mathbb{E}_X[D_{KL}[Q(Z|X)||P(Z)]] = \tag{15}$$

$$= \int\int p_D(x)q(z|x)\log\frac{q(z|x)}{p(z)}dzdx \tag{16}$$

$$= \int\int p_D(x)q(z|x)\log\frac{q(z|x)q(z)}{p(z)q(z)}dzdx \tag{17}$$

$$= \int\int p_D(x)q(z|x)\left(\log\frac{q(z|x)}{q(z)} + \log\frac{q(z)}{p(z)}\right)dzdx \tag{18}$$

$$= I[X,Z] + D_{KL}[Q(Z)||P(Z)] \tag{19}$$

Here, I multiplied by one using $q(z) = \int_{\mathcal{X}} q(z|x)p_D(x)dx$ The expectation of the KL-divergence term can be split up into the mutual information of X and Z and another KL-divergence between $Q(Z)$ and $P(Z)$:

$$\mathbb{E}_X[D_{KL}[Q(Z|X)||P(Z)]] = \underbrace{I[X,Z]}_{\text{shared information}} + \underbrace{D_{KL}[Q(Z)||P(Z)]}_{\text{coding inefficiency}}$$

This decomposition also uncovers two different things:

- The mutual information measure the number of bits that the decoder can use to decode X. If $I[X;Z]$ is small more blurry examples would be expected whereas for large values, the samples should become sharper.
- The mismatch between $Q(Z)$ and $P(Z)$ which measures how many bits are lost due to not using the informational bottleneck optimally.
- In theory, both terms can be optimized independently. There is no theoretic reason to not have high mutual information and low coding inefficiency. However, in practice, as neural networks parameterize $Q(Z|X)$, it can be expected that a high mutual information will increase the coding inefficiencies.

4.2.2 Comparision with Channel Capacity

I have derived two decompositions of mutual information: Frist using channel capacity: $I[X,Z] = C - J[Z]$ and in the previous section from the D_{KL} term in VAE: $I[X,Z] = \mathbb{E}_X[D_{KL}[Q(Z|X)||P(Z)]] - D_{KL}[Q(Z)||P(Z)]$. When the prior is chosen as Gaussian normal distribution, they are closely related.

For normal distributions, the KL-divergence can be calculated in closed form:

$$D_{KL}[Q_(Z|X)||\mathcal{N}(0,I)] = \frac{1}{2}\left(\text{tr}(\Sigma_X) + \mu_X^T\mu_X - k + \log\det(\Sigma_X)\right), \tag{20}$$

where μ_X and Σ_X are the predicted mean and the covariance matrix.

For this analysis, I assume that the mean and variance of $Q(Z)$ matches $P(Z)$. If this is not the case, batch-normalization could be used to make them match. I further require w.l.o.g that the variances of f_μ and f_σ are the same in all dimensions. Formally, the assumptions are:

$$\forall i : \text{Var}[f_\mu^{(i)}(X)] = s^2 \tag{21}$$

$$\forall i : \text{Var}[f_\sigma^{(i)}(X)] = n^2 \tag{22}$$

$$\mathbb{E}[Z] = 0 \tag{23}$$

$$\text{Var}[Z] = s^2 + n^2 = 1 \tag{24}$$

Using $\mathbb{E}_X[\text{tr}(\Sigma_X)] = kn^2$ and $\det(f_\sigma(X)) = \prod_{i=1}^k f_\sigma^{(i)}(X)^2$ as substitution for eq. (20) gives:

$$
\begin{aligned}
&\mathbb{E}_X[D_{KL}[Q_\phi(Z|X)||P(Z)]] = \\
&= \frac{1}{2}\left(\mathbb{E}_X\left[\text{tr}(\Sigma_X)\right]) + \mathbb{E}_X\left[f_\mu(X)^T f_\mu(X)\right] - k - \mathbb{E}_X\left[\log(\det(\Sigma_X))\right]\right) \\
&= \frac{1}{2}\left(kn^2 + \sum_{i=1}^{k} \underbrace{\mathbb{E}_X\left[f_\mu^{(i)}(X)^2\right]}_{=s^2} - k - \mathbb{E}_X\left[\log \prod_{i=1}^{k} f_\sigma^{(i)}(X)^2\right]\right) \\
&= \frac{1}{2}\left(kn^2 + ks^2 - k - \sum_{i=1}^{k} \mathbb{E}_X\left[\log f_\sigma^{(i)}(X)^2\right]\right) \\
&= \frac{1}{2}\left(k(\underbrace{n^2 + s^2}_{=1}) - k - \sum_{i=1}^{k} \mathbb{E}_X\left[\log f_\sigma^{(i)}(X)^2\right]\right) \\
&= -\frac{1}{2}\sum_{i=1}^{k} \mathbb{E}_X\left[\log f_\sigma^{(i)}(X)^2\right]
\end{aligned}
$$

Given that the mean and variances of $Q(Z)$ and $P(Z)$ match, the intensity of the noise fully determines the KL-term. This quantity can be further upper-bounded by using the Jenson's inequality:

$$
\begin{aligned}
&\leq -\frac{1}{2}\sum_{i=1}^{k} \log \underbrace{\mathbb{E}_X\left[f_\sigma^{(i)}(X)^2\right]}_{\text{Var}[f_\sigma^{(i)}(X)]=n^2} \\
&= \frac{k}{2}\log\left(\frac{1}{n^2}\right) \\
&= \frac{k}{2}\log\left(\frac{s^2 + n^s}{n^2}\right) \\
&= \frac{k}{2}\log\left(1 + \frac{s^2}{n^2}\right)
\end{aligned}
$$

Interestingly, the upper-bound using the Jenson's inequality gives exactly the channel capacity.

4.2.3 Coding inefficiencies of a Variational Autoencoder

Variational autoencoders have two inefficiencies: not using the full capacity of the information bottleneck and using an approximate decoder. These inefficiencies can be uncovered by applying Bayes' rule to the left KL-term of eq. (14) and rearranging the terms:

$$
\begin{aligned}
&\mathbb{E}_{X\sim P_D}[D_{KL}[Q(Z|X)||P(Z|X)]] = \\
&= \int\int p_D(x)q(z|x)\log\frac{q(z|x)}{p(z|x)}dzdx \\
&= \int\int p_D(x)q(z|x)\left(\log q(z|x) - \log p(z|x)\right)dzdx \\
&= \int\int q(x|z)q(z)\left(\log\frac{q(x|z)q(z)}{p_D(x)} - \log\frac{p(x|z)p(z)}{p(x)}\right)dzdx \\
&= \int\int q(x|z)q(z)\left(\log\frac{q(x|z)}{p(x|z)} + \log\frac{q(z)}{p(z)} - \log\frac{p_D(x)}{p(x)}\right)dzdx \\
&= \mathbb{E}_{X\sim P_D}[D_{KL}[Q(X|Z)||P(X|Z)]] + D_{KL}[Q(Z)||P(Z)] - D_{KL}[P_D(X)||P(X)]
\end{aligned}
\tag{25}
$$

Substituting this result back into eq. (14) and using the decomposition of the KL-divergence into entropy and cross entropy, i.e. $D_{KL}[P_D(X)||P(X)] = \mathbb{E}_{X\sim P_D}[-\log p(x)] - H[X]$, gives:

$$
\begin{aligned}
H[X] + \underbrace{\mathbb{E}_{X\sim P_D}[D_{KL}[Q(X|Z)||P(X|Z)]]}_{①} + \underbrace{D_{KL}[Q(Z)||P(Z)]}_{②} \\
= \mathbb{E}_{X,Z\sim Q(Z|X)}[-\log p_\theta(X|Z)] + \mathbb{E}_X[D_{KL}[Q(Z|X)||P(Z)]],
\end{aligned}
\tag{26}
$$

where $H[X]$ is the entropy of the data. This equation reveals the two inefficiencies ① and ②. The first term measures the derivation between the perfect decoder distribution $Q(X|Z)$ and the actual decoder distribution $P(X|Z)$. The second term quantifies how inefficient the encoder uses the information bottleneck. As the entropy term is fixed, minimizing the RHS will directly reduce the coding inefficiencies.

4.2.4 Information stored in the Latents of a Variational Autoencoder

In a VAE, not all information is stored in the latent variables. The information is rather splitted between the latent variables and the decoder probability distribution.

In the previous section, I argued that for a fixed capacity the information most useful for matching the data distribution is selected. For capacity autoencoders, the information most needed by the decoder could be controlled through the independence structure.

The analysis presented here is closely related to the (Chen et al., 2016) paper that analyses VAE using the Bits-Back Coding theory and also proposed to change the dependency structure of the encoder.

For VAEs, the transmitted information is determined by the optimization process. Why does a VAE not just transmit all information? A reason for this is the coding inefficiencies. A VAE will suffer coding inefficiencies ① & ② due to not powerful enough networks and suboptimal optimization. While there are in principle no limitations to the information

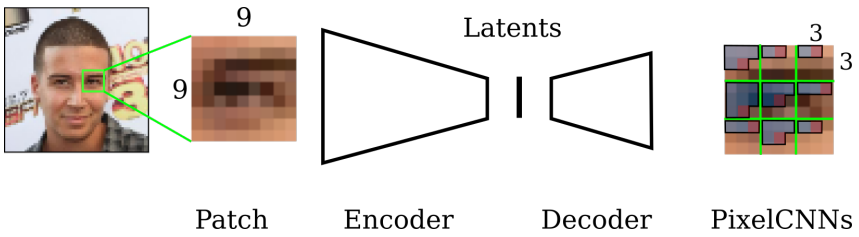

Figure 3: Overview of the patch-VAE. Patches of size 9x9 are sampled uniformly from the dataset. The VAE is trained on reconstructing the patches. In this figure, the VAE decoder distribution assumes independence of 3x3 patches given the representation. The dependencies within as 3x3 patch are modeled by small PixelCNNs models. Depending on the patch size, I experiment with independence sizes of 1x1, 3x3, and 5x5.

transmitted, the occurring coding inefficiencies create a barrier to the amount of information stored in the latent variables. There are two forces. First, the usefulness of information to the decoder which encourages to put more information into the latent code. Second, the amount of coding inefficencies suffered from transmitting information. It can be expected that transmitting less information will decreases the coding inefficiencies. So these two forces point in two different directions. Depending on the decoder distribution, different information will be selected.

If the decoder distribution can model any dependencies, the coding inefficiencies outweigh the usefulness. Any information transmitted could also be directly learned by the decoder without any coding inefficiencies and therefore the latent code will not be used at all.

However, if the decoder distribution is fully factorized, then a lot of information in Z is needed to match the factorized distribution to the true data distribution. As for the fixed capacity autoencoder, through the decoding independence structure determines which information is stored in the latent code.

The recently proposed β-VAE Higgins et al., 2016 introduces an additional hyperparameter to weight the KL-divergence more.

$$\mathbb{E}_{Z\sim Q(Z|X)}[\log p_\theta(X|Z)] - \beta D_{KL}[Q(Z|X)||P(Z)] \tag{27}$$

β is usually picked quite large, e.g. for CelebA $\beta = 250$. Each bit stored in Z has to provide now at least β bits in reconstruction. Of course, this leads to less bits stored in Z which results in worse reconstructions but the latent space also has some form of disentanglement.

5 Patch Variational Autoencoder

I want to capture the local structure at different scales, e.g. from edges over eyes to whole faces. Therefore, I use different patch sizes of 3x3, 9x9 and 15x15. As I discussed in the last section, the dependency structure of the decoder plays an important role on which information is stored in the latent code. Hence, I experimented with the following independence

Table 2: The mean maximum information ($M = \mathbb{E}_X[D_{KL}[Q(Z|X)||P(Z)]]$) stored in the latent code. A larger independency size reduces the information stored. For example, the 9|3 model stores less information than the 9|1 models.

Model	M [bits]	$\frac{M}{\#\text{pixel}}$ [bits]
β-VAE_256d	7.52	0.002
9|3	10.50	0.130
β-VAE_32d	11.62	0.003
3|1	13.44	1.493
15|5	13.83	0.061
15|3	32.02	0.142
9|1	32.71	0.404
15|1	74.93	0.333
VAE_32d	173.90	0.042
VAE_256d	622.83	0.152
VAE_4096d	830.37	0.203

(a) CelebA

Model	M [bits]	$\frac{M}{\#\text{pixel}}$ [bits]
9|3	9.83	0.121
15|5	11.59	0.052
3|1	12.04	1.337
15|3	25.76	0.114
9|1	38.67	0.477
15|1	57.51	0.256
VAE_32d	175.01	0.043
VAE_256d	589.11	0.144
VAE_4096d	811.42	0.198

(b) CIFAR-10

assumptions. Given the representation z, all pixels, 3x3, or 5x5 patches are independent. The local dependencies are modeled using simple PixelCNN models. The code used to refer to the patch-VAE models is `{patch size}|{independence size}`, e.g. 15|3 refers to the model that has a patch size of 15x15 and assumes all 3x3 patches are independent given z.

Table 3: The cross entropy on the test set. Using larger independence sizes decreases the cross entropy as the PixelCNNs can model the information more efficiently.

(a) CelebA

Model	**Cross Entropy [bits/dim]**
15|5	4.11
15|3	4.15
9|3	4.17
3|1	4.84
15|1	4.94
9|1	5.08

(b) CIFAR-10

Model	**Cross Entropy [bits/dim]**
15|5	4.33
9|3	4.60
15|3	4.60
3|1	5.31
9|1	5.46
15|1	5.64

The effect of the independence structure can be seen nicely when looking at the amount of information stored in the latent variables (shown in Table 2). Using a larger independence size decreases the information stored in the latent code. For example, the 9|3 model stores 10.50 bits in the mean whereas the 9|1 model uses 32.71 bits. This makes sense as local dependencies can be modeled by the PixelCNN directly and are not stored in the latent code. Due to the coding inefficiencies of the VAEs, it is favorable to model information locally by the PixelCNN.

In the next section, I describe how the experiments are set up to gain further knowledge about what information is stored in the latent code and how it compares to different baseline models.

6 Experiments

A useful representation should simplify the learning of a supervised task. To evaluate this, I propose to use the performance of a linear classifier that predicts the labels using only the unsupervised representation. If the representation is disentangled, a linear classifier should be sufficient to obtain good classification performance. The advantage of fitting a linear classifier is that it measures directly how suitable the learned representations are for a supervised task. Clearly, a disadvantage is that to evaluate the unsupervised model labels must be available. Furthermore, learning the classifier depends on hyperparameters such as the learning rate and the optimizer which could make it problematic to compare results between different work.

Data: The models are trained on the aligned CelebA dataset (Liu et al., 2015) and on CIFAR-10. The CelebA dataset is a common dataset for unsupervised models (Higgins et al., 2016; Dumoulin et al., 2016; Bachman, 2016). CelebA contains 40 binary face attributes, for example gender, different hair colors, facial expressions such as smiling or an open mouth. CelebA also contains landmark annotations for eyes, nose, and mouth. As common in the unsupervised literature, the CelebA images are preprocessed by a rectangular center cut and rescaled to a resolution of 64x64. The CIFAR-10 dataset contains 10 classes, e.g. airplane, cars. No preprocessing is done for the CIFAR-10 images.

Experiment: The models are trained in an unsupervised fashion without access to the labels. For training the linear classifiers, the representation of the patch VAEs are computed for all image patches, i.e. similar to convolution with stride 1.

$$p(y|x) = \text{sigmoid}(W f(x) + b), \tag{28}$$

where W, b are the learned parameters and the representation $f(x)$ is fixed. The training objective is binary cross entropy:

$$\max_{W,b} \mathbb{E}_{X \sim P_D(X)}[\log p(Y|X)] \tag{29}$$

The expectation is approximated using mini-batches of size 100. All linear classifiers are trained until convergence using normal SGD with a learning rate of 0.01.

Unsupervised Baselines: I compare the local patch VAEs to global VAEs and β-VAEs. As global VAE network architecture, I use the same as Higgins et al., 2016 and set $\beta = 250$. For CIFAR-10, only global VAEs are trained. I use different latent dimensions (32, 256, 4096) for the VAE to ensure that the latent dimension is not responsible for the reduced prediction performance.

Supervised Baselines CelebA: As a simple lower baseline, I trained a logistic regression from pixel space to face attributes (PIXEL).

For an upper baseline, deep neural networks are trained fully in a supervised fashion. As the CelebA dataset is preprocessed differently in the related work, results are not always comparable between them. In the unsupervised literature, it is common to use the aligned

images resized to 64x64. Whereas for supervised learning, the input images are at the full resolution of 178x218 or sometimes even upscaled to match ImageNet resolution.

Liu et al., 2015 who introduced the CelebA dataset showed accuracies only on the non-aligned images, which is a harder task. Their accuracy score of 87% is therefore of limited comparability.

The MOON paper by Rudd et al., 2016 adopted a VGG-16 network on the CelebA dataset. It uses the aligned images at an input resolution of 178x218 and introduces a method to balance the labels of the CelebA dataset. They obtained a mean accuracy performance of 90.94%. Günther et al., 2017 utilizes a ResNet-50 pretrained on ImageNet and adapted it to the face prediction task. Using an ensemble of 3 ResNets, they achieve 92.00%. As they use a pretrained network, it is not comparable to a method trained solely on CelebA.

A direct comparable supervised baseline should be trained on exactly the same images as the patch-VAEs, i.e. aligned and at a resolution of 64x64. Therefore, I trained a ResNet-18 on the same rescaled and aligned images. For the training of the ResNet-18, I augmented the data with color jittering, random scaling, rotation and horizontal flips (see Appendix 9.1 for the parameters). The test accuracy is with 90.42 % only slightly worse than Rudd et al., 2016 reported. The reduced image resolution could be a reason for this drop.

As the annotations in CelebA are unbalanced, the mean accuracy for making predictions at chance is already at around 80%. For some attributes e.g. bald, the chance accuracy is even higher than 95%. I included this baseline as CHANCE.

Supervised Baselines CIFAR-10: The current best model by (Real et al., 2018) achieves nearly 98% of accuracy. Karpathy, 2011 classified CIFAR-10 by hand and obtain an accuracy of around 94%. As done for CelebA, I trained a linear classifier from pixel space to labels (PIXEL).

7 Results

7.1 Reconstructions

Figure 4 and 5 shows reconstructed samples of the different VAE models. All samples are drawn from the model reconstruction probability distribution $p_\theta(x|z)$. For the β-VAE and VAE, not the predicted means but rather samples from their Gaussian reconstruction distribution are displayed. For the patch VAEs, non-overlapping patches are extracted and reconstructed individually.

As expected, the β-VAE shows high variances because its representation contains less information than the standard VAE. The reconstructions for the patch-VAEs 3|1, 9|1, and 15|1 are close to the original image. However, in the 15|1 details of the eyes are lost. Increasing the independence to larger patches as done in 9|3, 15|3, and 15|5, leads to more abstract representations. While the faces are barely visible for 9|3 and 15|5, the 15|3 model has preserved more information. The failure to reconstruct the images accurately corresponds to

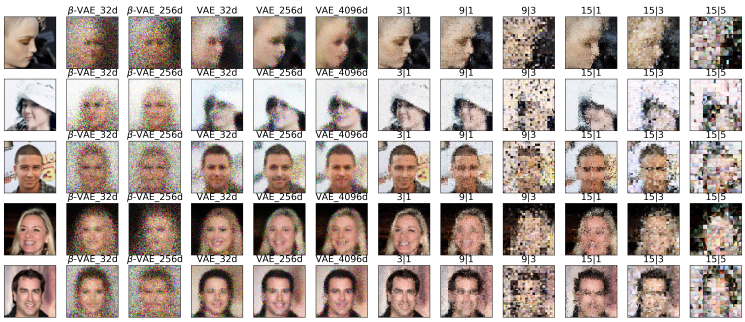

Figure 4: Reconstructed images for CelebA.

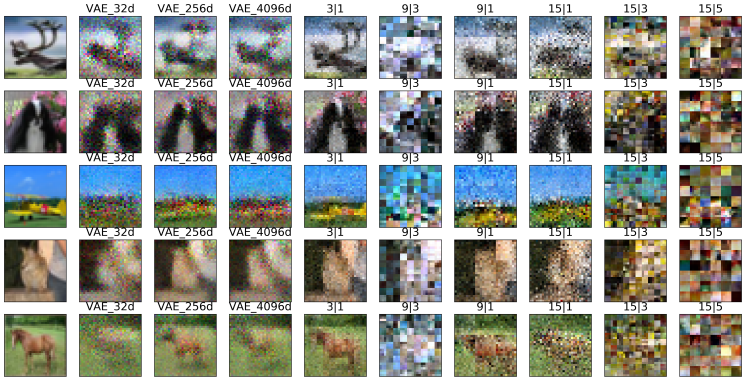

Figure 5: Reconstructed images for CIFAR-10.

the reduced information of the latent code which is shown in Table 2.

As pointed out by Theis and Bethge, 2015, the log-likelihood scores and visual quality of samples do not have to agree. As displayed in Table 3, the 15|5 model achieves the best log-likelihoods on both CIFAR-10 and CelebA but its samples look more like abstract art then the input images.

7.2 Similarities

The latent codes of the patch-VAE can be used as to find similar areas. The similarities are calculated using the L2-norm of the patch representations. All image patches are processed by the patch-VAEs such that I obtain a feature map of representations. Compared to the pmi measure, the similarities can be computed very fast. In Figure 6, the similarity score of the right eye is shown. For each patch-VAE, the similarity mapping looks differently. Both

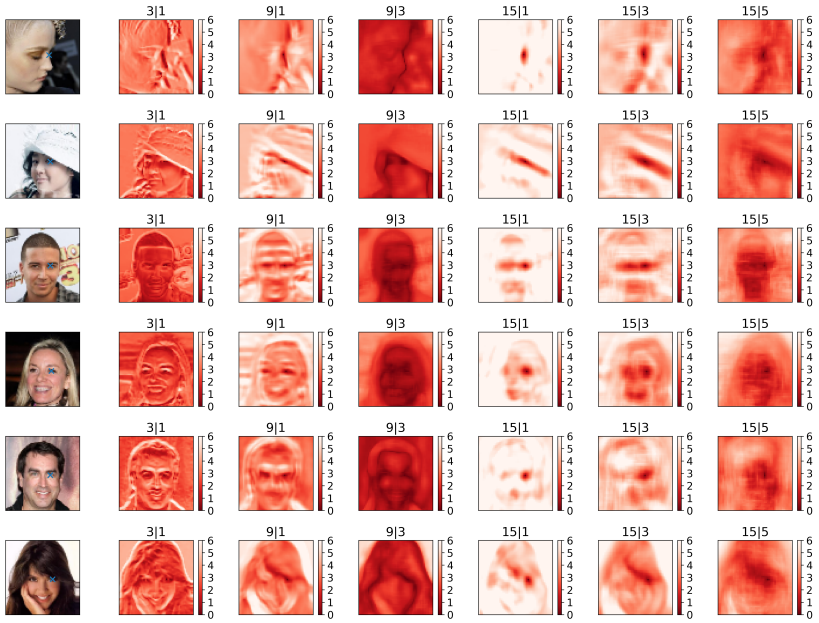

Figure 6: Similarities of the patch-VAEs for the right eye

patch size and the size of the independent patches matter. The small 3|1 model is sensitive for edges and colors. The more highlevel 9|1 model assigns similar scores to both eyes. Using large independence patches as in 9|3 and 15|5 models does not result in interesting similarity scores. As the 15|1 model has to store mostly all information, its similarity maps are very specific. In contrast, for the 15|3 model, the eye areas of some images are scored similar again.

7.3 Linear Classification Performance

Table 4 displays the performance of the supervised models and the linear classifier trained on the unsupervised representation. For both CIFAR-10 and CelebA, the best predictions are obtained when combining the representations of all models (ALL).

CelebA: The performance of ALL is only 1.25% from the supervised ResNet-18 model away. The best single patch-VAE is the 15|1 model closely followed by the 15|3 model. The linear classifiers trained on the representations of VAE and β-VAE score worse than the PIXEL baseline.

To test whether intermediate layers of the global VAE models contain are more useful features, I trained linear classifiers on them. The results are shown in Figure 11. Only in the first layers features with more usable information exists.

The accuracies of each patch-VAE for each attribute is shown in Figure 8. For many attributes the differences in accuracy between the supervised ResNet-18 and the PIXEL base-

Figure 7: Similarities of the patch-VAEs for hair

Table 4: Mean accuracy of the different models. Models listed on the top were trained in supervised fashion.

Model	**Accuracy [%]**
Günther et al., 2017 (ensemble, pretrained, 224x224)	92.00
Rudd et al., 2016 (aligned, 218x178)	90.96
ResNet-18 (aligned, 64x64)	90.42
Liu et al., 2015 (non-aligned)	87
ALL	89.17
15|1	88.63
15|3	88.58
9|1	88.23
15|5	88.20
3|1	87.97
9|3	86.98
PIXEL	86.52
VAE_4096d	86.26
VAE_256d	86.21
VAE_32d	84.38
β-VAE_32d	82.12
β-VAE_256d	81.74
CHANCE	80.04

(a) CelebA

Model	**Accuracy [%]**
Real et al., 2018	97.87
Human Karpathy, 2011	$\sim$ 94
ALL	61.48
15|5	49.16
15|3	49.08
9|1	48.71
15|1	48.21
9|3	47.92
VAE_256d	45.30
VAE_4096d	43.80
3|1	42.19
PIXEL	38.22
VAE_32d	37.66
CHANCE	10.00

(b) CIFAR-10

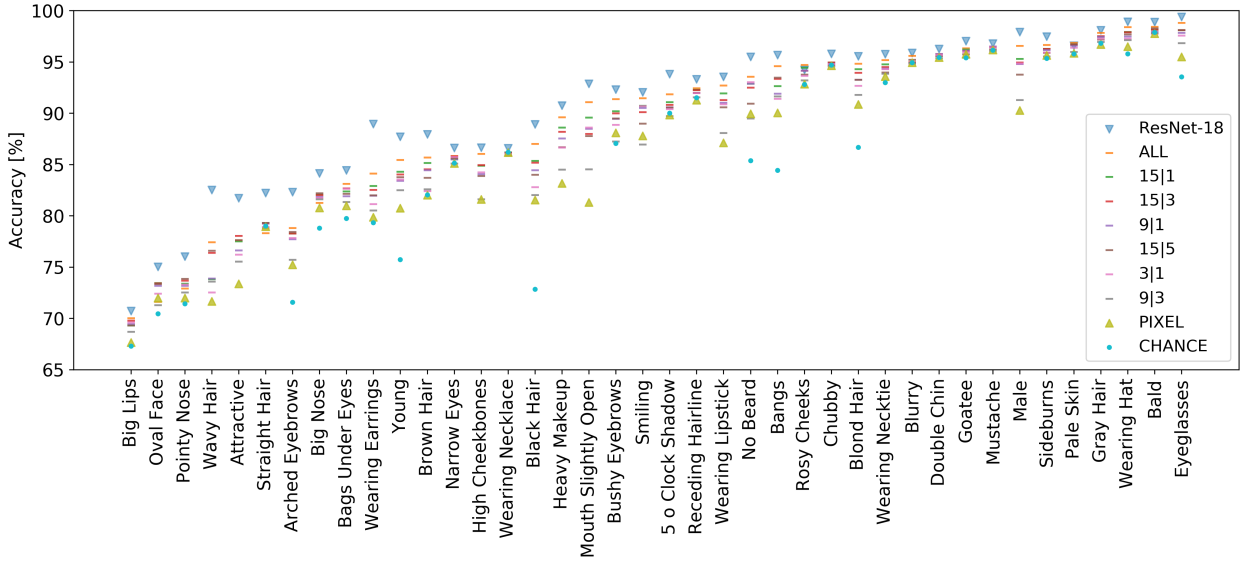

Figure 8: The accuracies of the patch-VAE models per attribute. CHANCE is the frequency the attributes appear, i.e. the accuracy that is achievalbe without looking at the image.

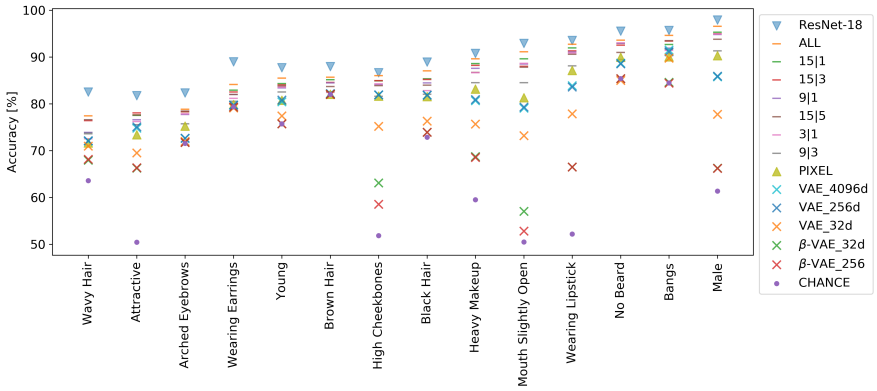

Figure 9: The accuracy for attributes where the difference between ResNet-18 and PIXEL is larger than 5%. Lines indicate patch-VAEs and crosses global VAE models.

line is small. Hence, I excluded all attributes with a difference smaller than 5% in Figure 9. While the patch-VAEs score consistently well for all attributes, the β-VAE models are close to chance for many attributes which is surprising as the β-VAE claims that the latent variables allow linear interpolation for attributes like gender.

CIFAR-10: The large difference compared to CelebA is that the performance of unsupervised models and supervised models are more far apart. The ALL model achieved about 35% less accuracy than the supervised baseline. We can see here again that the patch-VAE models have a higher accuracy than the latent representations of the global VAEs. However,

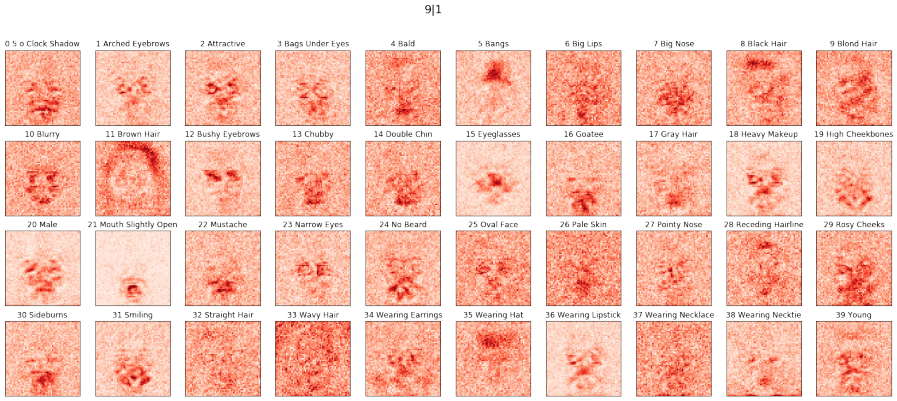

Figure 10: Weights of the logistic regression per attribute for the 9|1 model.

the representation of the global VAE_256d and VAE_4096d now performs better than the PIXEL baseline.

7.4 Weight inspection for CelebA

As I used the aligned faces in the CelebA data set, the weights of the linear classifier can be visualised. A larger positive weight corresponds to a large influence. Therefore, I displayed the positive parts of the weights in Figure 10, i.e.:

$$i(x, y) = \sum_{j=1}^{c} \max(0, w_{jxy}) \tag{30}$$

where $i(x, y)$ is the intensity at location x, y and the weight vector w is viewed as a 3-dimensional tensor with channels, height, width.

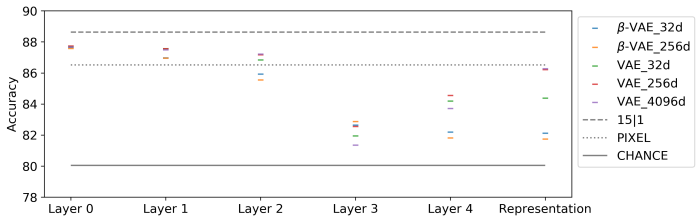

Figure 11: CelebA: Accuracies of linear classifiers trained on the features from the batch-norm layers of the VAE and β-VAE .

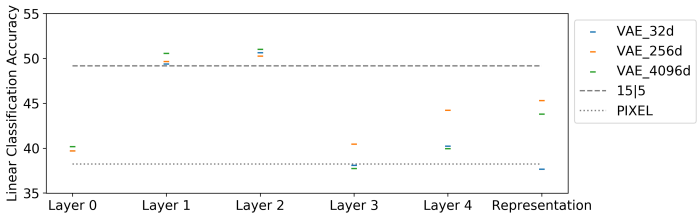

Figure 12: CIFAR-10: Accuracies of linear classifiers trained on the features from the batchnorm layers of the VAE.

8 Discussion

In my master thesis, I first showed an approach to map area dependencies explicitly using unsupervised learning. I identified multiple obstacles of mapping area dependencies explicitly. Merging areas based on their mutual information did not result in visually right looking areas. It would be interesting to explore attribution mechanism to find the areas which contain the most information of a given area.

In a detailed analysis, I related VAEs to the simple additive white Gaussian noise channel and uncovered its two coding efficiencies: not using the channel capacity fully and having an imperfect decoder. I continued to show which information is stored within the latent code of a VAE and how the information content can be controlled.

Building on the analysis of the VAEs, I introduced a local variant of VAEs that operates on image patches. I introduce a method to calculate similarities between patches and show how the different patch size and independence structure result in different similarities.

As an evaluation, I trained linear models on the learned representations. The performance of linear models give a good proxy for how useful the learned representations are for a supervised task. I found that for the CelebA dataset the performance using patch-VAEs is close to the fully supervised baseline whereas the representations of global VAEs are worse as features for the logistic regression than pure pixels.

A reason for the good performance of the patch-VAE models on CelebA could be that the face attributes of CelebA are local. For example, the attribute open mouth is clearly almost always on in same position which makes it easy for the linear classifier to predict it from the local patch-VAE representations.

An interesting finding is also that the feature maps of the first layers of the VAEs are a better input to the linear models than the last layers and the latent representations.

The results on the CIFAR-10 dataset show that unsupervised models still have a long way before them to catch up with supervised models.

In future work, it would be interesting to see the performance of patch-VAEs on other datasets, e.g. on ImageNet. Another line of future work would be to experiment with hierarchical VAE models in the evaluation.

References

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein gan". In: *arXiv preprint arXiv:1701.07875* (2017).

[2] Philip Bachman. "An architecture for deep, hierarchical generative models". In: *Advances in Neural Information Processing Systems*. 2016, pp. 4826–4834.

[3] Xi Chen et al. "Variational Lossy Autoencoder". In: *arXiv:1611.02731 [cs, stat]* (Nov. 2016). URL: http://arxiv.org/abs/1611.02731.

[4] J. Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*. 2009.

[5] Vincent Dumoulin et al. "Adversarially learned inference". In: *arXiv preprint arXiv:1606.00704* (2016).

[6] Ian Goodfellow et al. "Generative adversarial nets". In: *Advances in neural information processing systems*. 2014, pp. 2672–2680.

[7] Manuel Günther, Andras Rozsa, and Terranee E Boult. "AFFACT: Alignment-free facial attribute classification technique". In: *Biometrics (IJCB), 2017 IEEE International Joint Conference on*. IEEE. 2017, pp. 90–99.

[8] Irina Higgins et al. "Early visual concept learning with unsupervised deep learning". In: *arXiv preprint arXiv:1606.05579* (2016).

[9] Andrej Karpathy. *Lessons learned from manually classifying CIFAR-10*. http://karpathy.github.io/2011/04/27/manually-classifying-cifar10/. Blog. 2011.

[10] Tero Karras et al. "Progressive growing of gans for improved quality, stability, and variation". In: *arXiv preprint arXiv:1710.10196* (2017).

[11] Diederik P Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: Ml (2013), pp. 1–14. URL: http://arxiv.org/abs/1312.6114.

[12] Alex Krizhevsky. "Learning multiple layers of features from tiny images". In: (2009).

[13] Jussi T Lindgren, Jarmo Hurri, and Aapo Hyvärinen. "Spatial dependencies between local luminance and contrast in natural images". In: *Journal of Vision* 8.12 (2008), pp. 6–6.

[14] Ziwei Liu et al. "Deep Learning Face Attributes in the Wild". In: *Proceedings of International Conference on Computer Vision (ICCV)*. 2015.

[15] Aaron van den Oord et al. "Conditional Image Generation with PixelCNN Decoders". In: *arXiv:1606.05328 [cs]* (June 2016). arXiv: 1606.05328 [cs].

[16] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. "Pixel Recurrent Neural Networks". In: *arXiv:1601.06759 [cs]* (Jan. 2016). arXiv: 1601.06759 [cs].

[17] Esteban Real et al. "Regularized Evolution for Image Classifier Architecture Search". In: *arXiv preprint arXiv:1802.01548* (2018).

[18] Ethan M Rudd, Manuel Günther, and Terrance E Boult. "Moon: A mixed objective optimization network for the recognition of facial attributes". In: *European Conference on Computer Vision*. Springer. 2016, pp. 19–35.

[19] Tim Salimans et al. "PixelCNN++: Improving the PixelCNN with Discretized Logistic Mixture Likelihood and Other Modifications". In: *arXiv:1701.05517 [cs, stat]* (Jan. 2017). arXiv: 1701.05517 [cs, stat].

[20] C. E. Shannon. "A Mathematical Theory of Communication". In: *Bell System Technical Journal* 27.3 (1948), pp. 379–423. ISSN: 1538-7305. DOI: 10.1002/j.1538-7305.1948.tb01338.x. URL: http://dx.doi.org/10.1002/j.1538-7305.1948.tb01338.x.

[21] Casper Kaae Sønderby et al. "Ladder variational autoencoders". In: *Advances in neural information processing systems*. 2016, pp. 3738–3746.

[22] Lucas Theis and Matthias Bethge. "Generative Image Modeling Using Spatial LSTMs". In: *arXiv:1506.03478 [cs, stat]* (June 2015). URL: http://arxiv.org/abs/1506.03478.

[23] Shengjia Zhao, Jiaming Song, and Stefano Ermon. "Learning hierarchical features from generative models". In: *arXiv preprint arXiv:1702.08396* (2017).

9 Appendix

9.1 Augmentation for ResNet-18

The following augmentation was used for training the ResNet-18.

```
from torchvision.transforms import *

aug = Compose([
    ColorJitter(brightness=0.3, contrast=0.3, saturation=0.3),
    RandomHorizontalFlip(),
    RandomRotation([-10, 10]),
    RandomResizedCrop(64, scale=(0.6, 1)),
    ToTensor()
])
```