STEERING VQ-VAE TO IMPROVE OOD ROBUSTNESS ON A MEDICAL IMAGING TASK

Olivia Wiles, Florian Stimberg, Dan A. Calian, Timothy Mann, Sven Gowal
DeepMind
{oawiles,stimberg,dancalian,timothymann,sgowal}@google.com

ABSTRACT

Deep learning has made remarkable progress in solving tasks on large scale, labelled datasets. However, performance degrades when the model is expected to generalize to out of distribution (OOD) samples. This challenge is compounded when samples follow a long tail distribution where there are only a few samples under a given property. We explore using steerable directions as a controllable data augmentation technique to improve robustness to these challenges. In brief, we find directions in latent space that correspond to a desired change in image space (e.g. manipulating the staining properties of a cell image). This allows the generation of diverse, semantically meaningful samples. We evaluate our approach on the CAMELYON dataset where the task is to determine if cell images from different hospitals with different staining properties contain tumor cells. On this dataset, using our steerable directions improves baseline performance by 4% in the OOD case (the hospital is unseen at training time) and 19% in the low data setting (only a small number of samples from the given hospital are seen at training time).

1 INTRODUCTION

In the deep learning paradigm, it has become evident that the more data, the better. To achieve state of the art performance on ImageNet, Dosovitskiy et al. (2020) train on 300M images. However, for many domains, obtaining more and more data is simply not practical, and obtaining balanced datasets is unrealistic. For example, Ghorbani et al. (2020) note that in medical imaging, obtaining high quality, labelled, balanced datasets requires expert annotations, which is costly and cannot be obtained automatically. Moreover, poor performance in these low data regions has profound real world consequences.

In order to deal with these problems, prior work has explored how to use generative modelling such as generative adversarial networks (GANs) or variational auto-encoders (VAEs) as introduced by Goodfellow et al. (2014) and Kingma & Welling (2013). One approach is to generate new data by conditioning on a label and using this to generate new samples. However, GANs are susceptible to mode collapse, limiting the utility of this approach (Ravuri & Vinyals, 2019). Wong & Kolter (2020) demonstrate how a conditional VAE can describe a set of semantic perturbations. This set can be leveraged in order to enforce robustness to that semantic set; however, it requires access to perturbed versions of an image, which is not practical in many real world scenarios. Image to image models trained using GANs have been used successfully in medical imagery as a data augmentation device (Bowles et al., 2018; Ghorbani et al., 2020). While this setup does improve performance of the models, it requires obtaining the segmentation maps and training an image to image model.

We leverage steerable directions in order to manipulate a given image according to a desired change. Given an image $\boldsymbol{I}_X$ with a label (whether there are tumor cells) and a property X (the hospital it was obtained from), we use the steerable directions to manipulate the image to generate $\bar{\boldsymbol{I}}_Y$ with the same label but a new property Y (a different hospital). Images from different hospitals vary due to staining differences which may arise, for example, from different dyes used to stain the slides (see section A.4 for examples). These manipulated images are then used as data augmentation to train a downstream classifier, which should be more robust to variations of the given property.

Jahanian et al. (2020) introduced steerable directions and demonstrated that linear directions in BigGAN's latent space correspond to meaningful directions in image space. However, their work

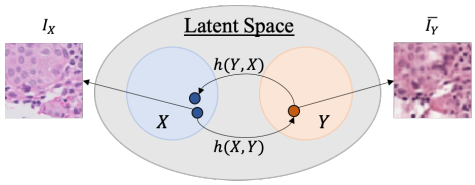


Figure 1: **Method overview.** Given a property X and a desired new property Y, we predict the steerable direction $h(X, Y)$ that transforms an image $\boldsymbol{I}_X$ of property X into one of Y while maintaining the label. We do this transformation in latent space, *not* image space.

has two limitations in our context. First, they do not operate on a given image but a sampled latent from BigGAN. We need to label the steered images in order to train the downstream classifier. As a result, we manipulate labelled images in the training set and so require the mapping from images to latents, but this mapping is non trivial in BigGAN. Second, they required paired data which we do not have access to in our setting.

We make the following contributions in order to generalize their approach to our setting. First, we extend steerable directions to a VQ-VAE model (van den Oord et al., 2017), which allows us to trivially find the latents corresponding to a given image. Second, we use cycle consistent training in order to overcome the need for paired data. Finally, instead of learning one direction in latent space for each change in image space, we learn one model that, given the current and desired property, predicts the corresponding direction in latent space.

Our approach has the following benefits. First, the steerable directions operate in the latent space of a generative model. This regularizes the generated samples to lie within the distribution of that generative model. If the generative model is trained well, this would correspond to the true image distribution. Second, we learn a mapping from latents to latents rather than an image to image model. As the latents are of much smaller dimension than the images, we can use a smaller model and less data. Finally, we can use the steerable directions to generate any number of samples for a given label with any combination of the given properties. We can use this to generate new samples for the properties that are within the tail of the distribution and re-balance the distribution of properties.

2 METHOD

We learn steerable directions on the latent space of a VQ-VAE model and use them to manipulate the labelled training images. These diverse images are then used to augment the training data of the downstream classifier.

2.1 STEERABILITY

Steerability using BigGAN. Steerability was introduced by Jahanian et al. (2020). They found that there exists linear directions in BigGAN's latent space that correspond to meaningful changes (such as zoom, or color changes) in image space. They find a given direction as follows. They randomly sample a latent $\boldsymbol{e}$ and corresponding generated image $\boldsymbol{I}_{\boldsymbol{e}}$ from BigGAN. They find the direction corresponding a given change as follows. For the image $\boldsymbol{I}_{\boldsymbol{e}}$, they apply the change (e.g. zoom) with some scalar severity $t \in [0, 1]$ to that image to create a manipulated image $\bar{\boldsymbol{I}}_{\boldsymbol{e}}$. They then minimize a reconstruction error to find a direction $\boldsymbol{d}$ such that the image $\boldsymbol{I}_g$ generated moving along the direction corresponds to the manipulated image: $\boldsymbol{I}_g = g(t \cdot \boldsymbol{d} + \boldsymbol{e}) \approx \bar{\boldsymbol{I}}_{\boldsymbol{e}}$ where g corresponds to passing the latents through the BigGAN model. Unfortunately, their approach cannot steer a given image (there is no simple mapping from image to BigGAN latent space) and requires being able to perturb the given image under the desired transformation.

Steerability using a VQ-VAE. We need to be able to find the latents that correspond to a given image. While Huh et al. (2020) demonstrate that a complex optimization procedure can be used to approximately find the corresponding latents of an image within a BigGAN model, it is slow and inaccurate. To overcome these downsides, we extend steerability to a VQ-VAE. We use a VQ-VAE as opposed to a VAE, as it is easy to train high quality decoders without "posterior collapse" (van den Oord et al., 2017). A VQ-VAE is trained in an auto-encoder fashion such that an image $\boldsymbol{I} \approx g(f(\boldsymbol{I}))$ where f is the encoder and g the decoder of the VQ-VAE. As a result, we can obtain the latents corresponding to a given image by passing it into the encoder: $\boldsymbol{e} = f(\boldsymbol{I})$.

Steerable directions without paired data. A second challenge of our setting is that we cannot assume paired data and do not want to learn a new direction for each pair of properties (for example,

Setup 1	ID VAL		OOD VAL		OOD TEST	
ResNet18	**97.1**	(0.7)	84.9	(1.2)	64.7	(3.8)
Ours (linear)	**96.7**	(0.6)	86.8	(0.7)	61.6	(2.4)
Ours (cond)	**96.6**	(0.6)	**87.8**	(2.5)	**70.8**	(2.5)

Setup 2	ID VAL		OOD VAL		OOD TEST	
ResNet18	**98.3**	(0.1)	80.0	(1.1)	59.9	(3.8)
Ours (linear)	97.2	(0.2)	**89.3**	(1.5)	61.4	(6.2)
Ours (cond)	97.1	(0.1)	87.5	(0.7)	**63.9**	(2.6)

Table 1: **Top 1 accuracy on the validation and test sets.** *Setup 1* uses a weight decay of 0.01 and averages the loss across the batch; *setup 2* uses a weight decay of 5e-4 and sums the loss over the batch. We report the mean and standard deviation (in parenthesis) and bold the best result in each column. For both training procedures, using our steerable directions as data augmentation improves OOD performance, while maintaining performance on the ID set.

a different model for hospital A to B and A to C). We overcome this as follows given an image $\boldsymbol{I}_X$ with property X and a desired new property Y (where $X, Y \in \mathbb{S}$ and $\mathbb{S}$ is the set of properties). We first obtain the embedding corresponding to $\boldsymbol{I}_X$, $\boldsymbol{e}_X = f(\boldsymbol{I}_X)$. The direction is conditioned on the two properties using a neural network h: $\boldsymbol{d}_{XY} = h(X, Y)$. Then $\boldsymbol{e}_Y = \boldsymbol{e}_X + \boldsymbol{d}_{XY}$ and finally $\bar{\boldsymbol{I}}_Y = g(\boldsymbol{e}_Y)$. This is the *linear* case. We also consider a variant, the *conditional* case, where $\boldsymbol{d}_{XY} = h(\boldsymbol{e}_X, X, Y)$. The *linear* case has less flexibility, so it can be trained with very little data, but the *conditional* case can describe more complex transformations in latent space.

2.2 TRAINING

Steerable directions. We use the following three loss functions to train $h(\cdot)$, while keeping the encoder and decoder of the generative model fixed. Full details are in the supplementary material.

First, a discriminative loss $\mathcal{L}_d(\boldsymbol{I}_X, \bar{\boldsymbol{I}}_Y)$ enforces that $\bar{\boldsymbol{I}}_Y$ is indistinguishable from a real image. Second, a reconstruction loss $\mathcal{L}_r(\boldsymbol{I}_X, X, Y)$ enforces that if we transform an image $\boldsymbol{I}_X$ to have property Y and obtain $\bar{\boldsymbol{I}}_Y$, then enforcing $\bar{\boldsymbol{I}}_Y$ to have property X should recover $\boldsymbol{I}_X$. The combination of a reconstruction and discriminative loss is inspired by the cycle consistency approach, introduced by Zhu et al. (2017) to learn transformations between sets of images. Finally, a classification loss $\mathcal{L}_c(\bar{\boldsymbol{I}}_Y)$ enforces that a classifier C that correctly classifies real images, classifies $\bar{\boldsymbol{I}}_Y$ as having property Y. The classifier is trained only on the real images, but its frozen weights are used to train the steerable directions. The final loss is $\mathcal{L} = 0.5\mathcal{L}_d(\boldsymbol{I}_X, \bar{\boldsymbol{I}}_Y) + 0.5\mathcal{L}_c(\bar{\boldsymbol{I}}_Y) + 20\mathcal{L}_r(\boldsymbol{I}_X, X, Y)$.

The downstream classifier. Given the trained weights of h for some set of properties, we augment the original dataset as follows when training the downstream classifier. We augment a batch of size N with another N samples which are then manipulated while keeping the true label. For a given sample $\boldsymbol{I}_X$ within the batch, we randomly select a new property $Z \in \mathbb{S}$ and how far along the direction to move $t \in [0, 1]$. We then generate a manipulated image $\bar{\boldsymbol{I}}_Z = g(f(\boldsymbol{I}_X) + t \cdot \boldsymbol{d}_{XZ})$. Visualisations of how t and Z impact the generated images are given in section A.5.

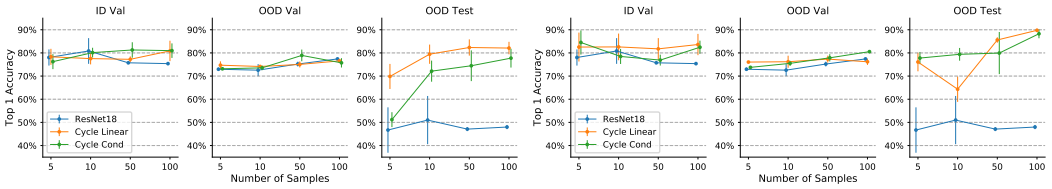
3 EXPERIMENTS

We evaluate our approach in two settings. The first setting is the out-of-distribution (OOD) setting in section 3.1. This setting evaluates how well the steerable directions improve the model's generalization capabilities. Augmenting the training set with multiple, diverse manipulations of each image should encourage the model to be more robust to these semantic manipulations.

The second setting is the low data setting in section 3.2. This setting evaluates whether the directions can be learned from only a few samples and whether these directions can be used to manipulate the training set to produce a more robust model.

Dataset. We use the CAMELYON dataset (Koh et al., 2020; Bandi et al., 2018). The data comes from five hospitals and the task is to classify images as having tumor cells or not. For training, three of the five hospitals are used. The other two are left for OOD validation and test. There is also an in-distribution (ID) validation set, which comes from the same three hospitals as the training dataset.

Steerable directions. We only use the hospital property to train the steerable directions. The VQ-VAE model is trained on the full training set (*without* labels). The steerable directions are trained for 2K iterations with a batch size of 32 and a learning rate of 10^{-4} and the Adam optimizer.



(a) VQ-VAE is trained on the full training set.

(b) VQ-VAE is trained on the low data subset.

Figure 2: **Low data setting.** We vary the number of samples N in the low data regions, creating new training sets. We re-train the classifiers, steerable directions, and (optionally) the VQ-VAE model on these subsampled training sets. We compare to a baseline (ResNet18), performing a bit better on ID and OOD val but much better on OOD test.

Training setup. All classification models are ResNet18 models (He et al., 2016) and trained to minimize the binary cross entropy loss. They are trained with SGD, momentum of 0.9, learning rate of 10^{-3}, weight decay of 10^{-2}, batch size of 1024, and for 900 epochs. For each experiment, we run the models with three seeds and report the mean and standard deviation.

3.1 OOD Setting

In table 1 we compare our data augmentation technique to using no data augmentation under two training setups. We also compare using the *linear* and *conditional* case. As can be seen, we outperform the baseline on both setups on both the val and test set. In this case, using the conditional steerable direction performs better than the linear ones. Moreover, our data augmentation procedure comes at minimal cost to ID performance.

3.2 Low Data Setting

To create low data regions, we modify the training set such that we use all samples from hospital 1 but only N from hospitals 2 and 3 each. This subset is used to train both the steerable directions and the downstream classifier. We optionally retrain the underlying VQ-VAE generative model on the same subset. When training the steerable directions and VQ-VAE, as we are simply transforming images (and not classifying), we augment the N samples by randomly cropping and resizing the images. Doing this when classifying could invalidate the tumor label, so this cropping and resizing operation is only performed when finding steerable directions.

In figure 2 we compare the performance using the steerable directions to directly using the N samples to train the classifier. Our approach (with and without retraining the VQ-VAE) performs consistently similarly or better than the baseline on the ID and OOD (val) errors. However, our setup (both the *conditional* and *linear* case) perform substantially better on the OOD test set. This may be due to the augmentation procedure and steerable directions encouraging better generalization. Additionally, given additional samples our model improves in all cases, whereas the baseline does not improve on OOD test.

We note that it seems the samples in the training set generalize better to OOD val than test (as demonstrated by table 1 – visualizations are given in section A.4). As a result, the steerable directions are more beneficial in the OOD test case, where there is a larger generalization gap. Using our data augmentation procedure removes this gap, giving similar performance on all data subsets.

4 Conclusion

This work has demonstrated how to use steerable directions as a form of data augmentation. This was done by extending steerable directions on BigGAN to a VQ-VAE setting and learning the directions on sets of images with different properties, as opposed to paired data. The steerable directions can be used to improve both OOD generalization and generalization when training with unbalanced data. It will be interesting to explore extending this approach to other datasets and generative models.

REFERENCES

Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.

Peter Bandi, Oscar Geessink, Quirine Manson, Marcory Van Dijk, Maschenka Balkenhol, Meyke Hermsen, Babak Ehteshami Bejnordi, Byungjae Lee, Kyunghyun Paeng, Aoxiao Zhong, et al. From detection of individual metastases to classification of lymph node status at the patient level: the CAMELYON17 challenge. *IEEE Transactions on Medical Imaging*, 2018.

Christopher Bowles, Liang Chen, Ricardo Guerrero, Paul Bentley, Roger Gunn, Alexander Hammers, David Alexander Dickie, Maria Valdés Hernández, Joanna Wardlaw, and Daniel Rueckert. GAN augmentation: Augmenting training data using generative adversarial networks. *arXiv preprint arXiv:1810.10863*, 2018.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Amirata Ghorbani, Vivek Natarajan, David Coz, and Yuan Liu. DermGAN: synthetic generation of clinical skin images with pathology. In *Machine Learning for Health Workshop*. PMLR, 2020.

Karan Goel, Albert Gu, Yixuan Li, and Christopher Ré. Model patching: Closing the subgroup performance gap with data augmentation. *arXiv preprint arXiv:2008.06775*, 2020.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2014.

Sven Gowal, Chongli Qin, Po-Sen Huang, Taylan Cemgil, Krishnamurthy Dvijotham, Timothy Mann, and Pushmeet Kohli. Achieving robustness in the wild via adversarial mixing with disentangled representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1211–1220, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Minyoung Huh, Richard Zhang, Jun-Yan Zhu, Sylvain Paris, and Aaron Hertzmann. Transforming and projecting images into class-conditional generative networks. In *Proceedings of the European Conference on Computer Vision*, 2020.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Ali Jahanian, Lucy Chai, and Phillip Isola. On the 'steerability' of generative adversarial networks. In *Proceedings of the International Conference on Learning Representations*, 2020.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Sara Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. WILDS: A benchmark of in-the-wild distribution shifts. *arXiv preprint arXiv:2012.07421*, 2020.

Giovanni Mariani, Florian Scheidegger, Roxana Istrate, Costas Bekas, and Cristiano Malossi. BAGAN: Data augmentation with balancing GAN. *arXiv preprint arXiv:1803.09655*, 2018.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the International Conference on Machine Learning*, 2010.

Suman Ravuri and Oriol Vinyals. Seeing is not necessarily believing: Limitations of BigGANs for data augmentation. In *Proceedings of the ICLR Workshop on Learning from Limited Labelled Data*, 2019.

Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks. In *Proceedings of the International Conference on Learning Representations*, 2019.

Nimit S Sohoni, Jared A Dunnmon, Geoffrey Angus, Albert Gu, and Christopher Ré. No subclass left behind: Fine-grained robustness in coarse-grained classification problems. In *Advances in Neural Information Processing Systems*, 2020.

Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, 2017.

Yaxing Wang, Chenshen Wu, Luis Herranz, Joost van de Weijer, Abel Gonzalez-Garcia, and Bogdan Raducanu. Transferring GANs: generating images from limited data. In *Proceedings of the European Conference on Computer Vision*, 2018.

Eric Wong and J Zico Kolter. Learning perturbation sets for robust machine learning. *arXiv preprint arXiv:2007.08450*, 2020.

Xiaofeng Zhang, Zhangyang Wang, Dong Liu, and Qing Ling. Dada: Deep adversarial data augmentation for extremely low data regime classification. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the International Conference on Computer Vision*, 2017.

A APPENDIX

A.1 RELATED WORK

We discuss the most relevant methods to our work here. In particular, we discuss methods that aim to improve robustness when datasets are imbalanced, with only a few samples from a given domain. One approach is to optimize for the worst-case performance (Sagawa et al., 2019; Sohoni et al., 2020). Another approach is to use data augmentation via GANs to rebalance and (or) augment the training distribution (Antoniou et al., 2017; Mariani et al., 2018; Wang et al., 2018; Zhang et al., 2019). However, these GANs are hard to train and have limited guarantees as to whether they will properly model the low data regions.

Most similar to ours are ADVMIX by Gowal et al. (2020) and CAMEL by Goel et al. (2020), which both aim to learn robustness to semantic perturbations. ADVMIX uses the style and content disentanglement within STYLEGAN (introduced by Karras et al. (2019)) to learn a representation robust to style changes. However, it does not generalize to other semantic changes or classes not well described by a STYLEGAN model. CAMEL uses CYCLEGAN (introduced by Zhu et al. (2017)) and additional consistency constraints to learn a model robust to a given set of changes. However, a new direction must be learned for each change (unlike our method which learns a model for a *set* of changes) and they do not use a generative model to regularize the learned directions and to take advantage of additional, unlabelled samples.

A.2 FURTHER MODEL DETAILS

A.2.1 DATASET

We use 64×64 images for all models.

A.2.2 VQ-VAE

We use a VQ-VAE with two residual layers, giving a latent space of size $16 \times 16 \times 64$ for the 64×64 images. The VQ-VAE embedding space (or the dictionary) consists of 768 embeddings of size 64. These dictionary embeddings are updated with exponential moving average (EMA). We extract the $16 \times 16 \times 64$ embeddings *before* passing them to the quantizer operation in the VQ-VAE when learning steerable directions.

A.2.3 STEERABLE DIRECTIONS

The steerable directions are implemented as MLPs.

Linear case. In the *linear* case, we use a 4 layer MLP that, given the two properties X and Y, predicts a direction $\boldsymbol{d}_{XY}$ such that adding $\boldsymbol{d}_{XY}$ to the latent space of an image with property X gives one with property Y. X and Y are encoded as one hot labels. The MLP has size $64 \to 64 \to 64 \to (16 \times 16 \times 64)$ and uses ReLU nonlinearities (Nair & Hinton, 2010). Note that the direction is fixed for each ordering of properties and could be precomputed.

Conditional case. In the *conditional* case, we use two MLPs. One projects the one hot encoding of the two properties X, Y to an embedding space $\boldsymbol{p}_{XY}$. This MLP has size $64 \to 64 \to 64$. The other takes the embedding of the two properties $\boldsymbol{p}_{XY}$ and the latents $\boldsymbol{e}_X$ corresponding to an image of property X. It predicts a direction $\boldsymbol{d}_{XY}$ to apply to the current set of latents $\boldsymbol{e}_X$. This MLP has size $64 \to 64 \to 64 \to (16 \times 16 \times 64)$.

A.3 FURTHER TRAINING DETAILS

A.3.1 TRAINING STEERABLE DIRECTIONS

Assume we have a batch of real images $\mathbf{I}$. For a given image in that batch $\boldsymbol{I}_X$ with property X, we randomly select a new property $Y \in \mathbb{S}$ where $\mathbb{S}$ is the set of properties and generate $\bar{\boldsymbol{I}}_Y$. $\bar{\boldsymbol{I}}_Y$ should have the same label as $\boldsymbol{I}_X$ but the property Y. To do this, we predict a direction $h(\cdot) = \boldsymbol{d}_{XY}$ such that $\bar{\boldsymbol{I}}_Y = f(\boldsymbol{I}_X) + \boldsymbol{d}_{XY}$. To train $h(\cdot)$ to find a meaningful direction in latent space, we use the following three losses.

Note that we train auxiliary networks (D and C) for the discriminative and classification losses, but these are unused at test time.

Discriminative loss. The discriminative loss $\mathcal{L}_d(\boldsymbol{I}_X, \bar{\boldsymbol{I}}_Y)$ enforces that $\bar{\boldsymbol{I}}_Y$ looks like a real image. We employ a patch based discriminator D in the standard min max fashion, as described by Isola et al. (2017). The discriminator aims to distinguish between real images $\boldsymbol{I}$ and fake images $\bar{\boldsymbol{I}}$ (by maximizing the objective) and the generator aims to fool it (by minimizing the objective).

$$\mathcal{L}_d(\boldsymbol{I}, \bar{\boldsymbol{I}}) = \mathbb{E}[\log D(\boldsymbol{I})] + \mathbb{E}[\log(1 - D(\bar{\boldsymbol{I}}))] \tag{1}$$

In this loss, we keep f and g (the encoder and decoder of the VQ-VAE model) fixed. We train $h(\cdot)$ and the patch based discriminator D.

Reconstruction loss. The reconstruction loss $\mathcal{L}_r(\boldsymbol{I}_X, X, Y)$ uses a simple mean squared error to enforce the consistency between $\boldsymbol{I}_X$ and the one generated by performing a cycle in latent space – transforming $\boldsymbol{I}_X$ to have property Y and then property X again.

$$\mathcal{L}_r(\boldsymbol{I}_X, X, Y) = ||\boldsymbol{I}_X - g((f \circ g)(\boldsymbol{e}_X + \boldsymbol{d}_{XY}) + \boldsymbol{d}_{YX})|| \tag{2}$$

In this loss, we keep f and g fixed. We train $h(\cdot)$ only.

Classification loss. The classification loss $\mathcal{L}_c(\bar{\boldsymbol{I}}_Y)$ enforces that $\bar{\boldsymbol{I}}_Y$ has property Y. A classifier is first trained on real images using a cross entropy loss. The classifier's frozen weights are used to train the steerable directions by enforcing that, when generating an image with a property Z, the classifier predicts the image as having property Z ($\log(C(\bar{\boldsymbol{I}}_Y)_Z)$ is the probability $\bar{\boldsymbol{I}}_Y$ has property Z).

$$\mathcal{L}_c(\bar{\boldsymbol{I}}_Y) = \mathbb{E}[\sum_{Z \in S} \mathbf{1}_{\mathrm{Z=Y}} \log(C(\bar{\boldsymbol{I}}_Y)_Z)] \tag{3}$$

In this loss, we train C. We then keep C fixed and only update $h(\cdot)$ when training the steerable directions.

A.3.2 Hyperparameters for the Steerable Directions

Training D and C. We use the same hyperparameters (optimizer, learning rate, batch size, and number of iterations) when training the auxiliary networks (D and C).

Training in the low data regime. In the low data regime, we used slightly different hyperparameters to train the steerable directions. These hyperparameters are kept constant when re-training the VQ-VAE and when using different numbers of samples.

For the linear case, we used a final loss of $\mathcal{L} = 0.1\mathcal{L}_d(\boldsymbol{I}_X, \bar{\boldsymbol{I}}_Y) + 0.1\mathcal{L}_c(\bar{\boldsymbol{I}}_Y) + 20\mathcal{L}_r(\boldsymbol{I}_X, X, Y)$ and learning rate of 5e-5.

For the conditional case, we used a final loss of $\mathcal{L} = 10\mathcal{L}_d(\boldsymbol{I}_X, \bar{\boldsymbol{I}}_Y) + 10\mathcal{L}_c(\bar{\boldsymbol{I}}_Y) + 20\mathcal{L}_r(\boldsymbol{I}_X, X, Y)$ and learning rate of 1e-4.

To find these hyperparamters, we first find the minimum weighting of $\mathcal{L}_r$ which recovers the original image. If the learning rate is too small, then the original image will not be recovered but if it is too large then the other losses will not be able to find steerable directions that transform the properties of the image. We then set the other hyperparameters while keeping the weighting for $\mathcal{L}_r$ fixed.

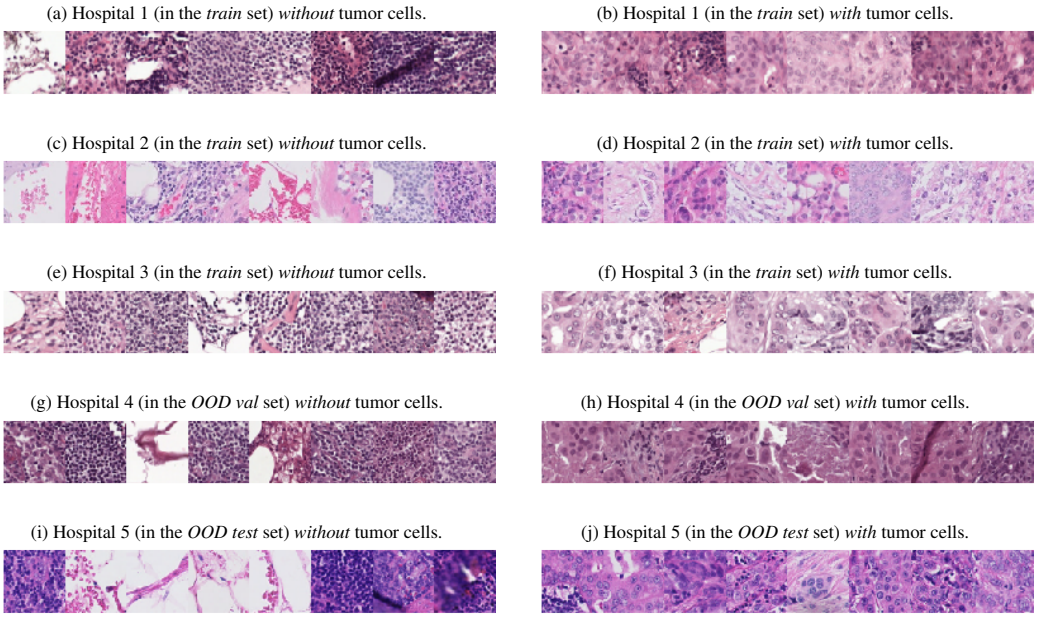
A.4 Visualization of Samples from each Data Distribution

We visualize samples from each of the data distributions: train/ID val, OOD val, and OOD test in figure 3. This figure visualizes the different staining properties of each hospital, as well as how the cell images vary under the different labels (with and without tumor cells). From this we can see that the staining properties differ substantially and impact the coloring of the cells and the amount of saturation and contrast in the images. Moreover, the staining properties of the OOD val images are nearer those of the training images than those of the OOD test images. This explains why there is a less of a performance gap between OOD val and ID test versus OOD test and ID test.

A.5 Learned Steerable Directions

We visualize the steerable directions obtained from our models trained in the *linear* and *conditional* cases in figure 4 and figure 5 respectively. From visualizing these directions, we can see marked

differences in the two cases. The linear case has less flexibility, as the predicted direction is a function of the two hospital properties – the original hospital the image comes from and the new hospital we wish to steer it towards. The conditional case has more flexibility, as the predicted direction is additionally conditioned on the latents corresponding to the given image. As a result, the generated images in the linear case add in fewer artefacts and structural changes than in the conditional case. This presumably explains the difference in performance between the two approaches.



(a) Hospital 1 (in the *train* set) *without* tumor cells.

(b) Hospital 1 (in the *train* set) *with* tumor cells.

(c) Hospital 2 (in the *train* set) *without* tumor cells.

(d) Hospital 2 (in the *train* set) *with* tumor cells.

(e) Hospital 3 (in the *train* set) *without* tumor cells.

(f) Hospital 3 (in the *train* set) *with* tumor cells.

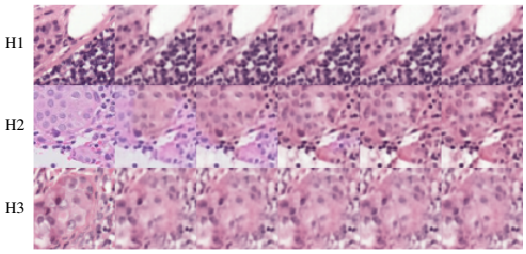
(g) Hospital 4 (in the *OOD val* set) *without* tumor cells.

(h) Hospital 4 (in the *OOD val* set) *with* tumor cells.

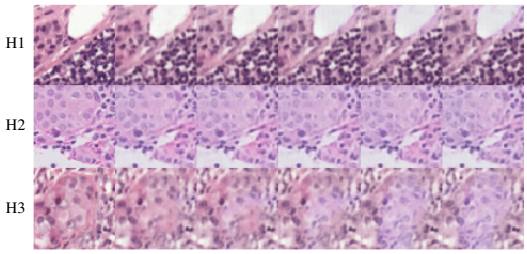
(i) Hospital 5 (in the *OOD test* set) *without* tumor cells.

(j) Hospital 5 (in the *OOD test* set) *with* tumor cells.

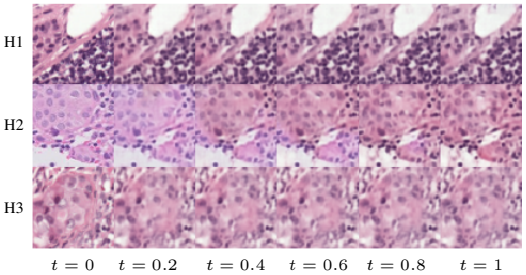
Figure 3: Samples from each of the hospitals with and without tumor cells. Tumor cells can be seen as the larger darker blobs in these images as opposed to the images without tumor cells. Note the different staining properties from each hospital. For example, hospital 2 has more contrast and saturation and more color than hospitals 1 and 3. Hospital 1 has more red tones whereas hospital 3 has more purple tones. These staining properties also differ substantially between the train set and the OOD val and test sets. Moreover, we can see that hospital 4 (OOD val) is more similar to hospital 1 (train) than hospital 5 (OOD test) is to any hospital. This explains why there is less of a performance gap between ID test and OOD val versus OOD test.



(a) Steering images from hospitals 1 (H1), 2 (H2) and 3 (H3) to come from H1.

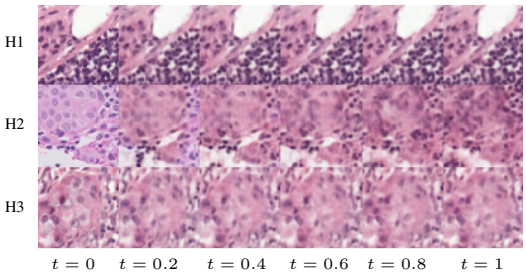


(b) Steering images from hospitals 1 (H1), 2 (H2) and 3 (H3) to come from H2.

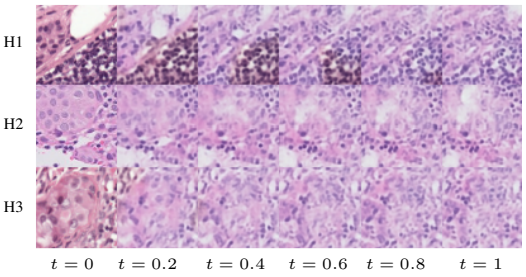


(c) Steering images from hospitals 1 (H1), 2 (H2) and 3 (H3) to come from H3.

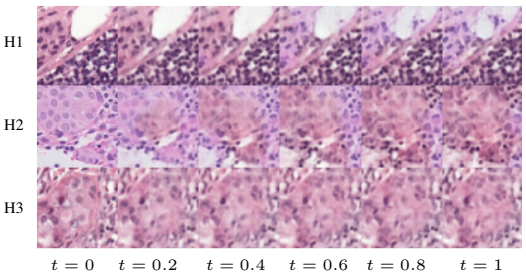
Figure 4: **Steerable directions for the *linear* model.** For three images, one from each of the training hospitals, we visualize how the steerable directions transform them. As can be seen, the high level semantics of the images are mostly preserved while the 'staining properties' (coloring of the slides) is manipulated.



(a) Steering images from hospitals 1 (H1), 2 (H2) and 3 (H3) to come from H1.



(b) Steering images from hospitals 1 (H1), 2 (H2) and 3 (H3) to come from H2.



(c) Steering images from hospitals 1 (H1), 2 (H2) and 3 (H3) to come from H3.

Figure 5: **Steerable directions for the *conditional* model.** For three images, one from each of the training hospitals, we visualize how the steerable directions transform them. As can be seen, these steerable directions, which have more flexibility, add more noise and artefacts to the original images.