Image Synthesis with a Convolutional Capsule Generative Adversarial Network

Cher Bass[1,2,3] C.BASS14@IMPERIAL.AC.UK
Tianhong Dai[1] TIANHONG.DAI15@IMPERIAL.AC.UK
Benjamin Billot[1] BENJAMIN.BILLOT.18@UCL.AC.UK
Kai Arulkumaran[1] KAILASH.ARULKUMARAN13@IMPERIAL.AC.UK
Antonia Creswell[1] ANTONIA.CRESWELL11@IMPERIAL.AC.UK
Claudia Clopath[1] C.CLOPATH@IMPERIAL.AC.UK
Vincenzo De Paola[3] VINCENZO.DEPAOLA@CSC.MRC.AC.UK
Anil Anthony Bharath[1] A.BHARATH@IMPERIAL.AC.UK

[1] *Department of Bioengineering, Imperial College London, UK*
[2] *Centre for Neurotechnology, Imperial College London, UK*
[3] *MRC Clinical Science Centre, Faculty of Medicine, Imperial College London, London, UK*

Editors: Under Review for MIDL 2019

Abstract

Machine learning for biomedical imaging often suffers from a lack of labelled training data. One solution is to use generative models to synthesise more data. To this end, we introduce CapsPix2Pix, which combines convolutional capsules with the `pix2pix` framework, to synthesise images conditioned on class segmentation labels. We apply our approach to a new biomedical dataset of cortical axons imaged by two-photon microscopy, as a method of data augmentation for small datasets. We evaluate performance both qualitatively and quantitatively. Quantitative evaluation is performed by using image data generated by either CapsPix2Pix or `pix2pix` to train a U-net on a segmentation task, then testing on real microscopy data. Our method quantitatively performs as well as `pix2pix`, with an order of magnitude fewer parameters. Additionally, CapsPix2Pix is far more capable at synthesising images of different appearance, but the same underlying geometry. Finally, qualitative analysis of the features learned by CapsPix2Pix suggests that individual capsules capture diverse and often semantically meaningful groups of features, covering structures such as synapses, axons and noise.

Keywords: Capsule Network, Generative Adversarial Network, Neurons, Axons, Synthetic Data, Segmentation, Image synthesis, Image-to-Image translation

1. Introduction

Deep neural networks (DNNs) have significantly advanced the state-of-the-art in biomedical image analysis, particularly where data is well curated for specific tasks such as segmentation and classification (Ronneberger et al., 2015; Frid-Adar et al., 2018b). However, there remain significant challenges. A key problem in analysing biomedical datasets is inadequate quantities of data for training. This may occur where data is scarce, or the expert resources

are needed for curated ground truth. We may specifically care about learning from a few data points, as animal or patient data is often restricted.

A way to resolve this is to train a generative model on modest amounts of labelled data, and using this to augment the dataset with synthesised images. To address this, we introduce a convolutional capsule generative adversarial network (GAN), CapsPix2Pix, to synthesise images conditioned on segmentation labels. Through experimental evaluation we show that our method quantitatively and qualitatively matches or outperforms `pix2pix`, a state-of-the-art conditional image synthesis model (Isola et al., 2017).

In particular, we use our method to synthesise images based on a new 152-image cortical axon dataset, captured with a two-photon microscope in the mouse cortex. We use the synthesised images to pretrain a segmentation model, and show that it improves performance over using only the original dataset. Better segmentation performance allows us to better automate the study of neurons. This is relevant for the field of experimental neuroscience, where there is interest in examining the structure of neurons under different conditions, or in response to a stimuli. We believe that our results validate the use of CapsPix2Pix, which could be applied to other biomedical datasets and downstream tasks.

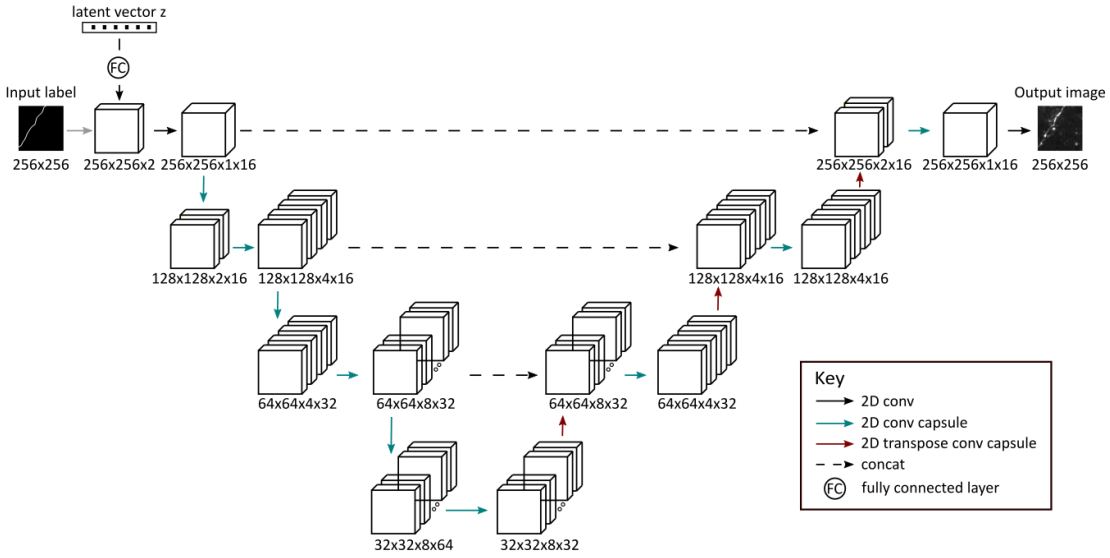

Figure 1: CapsPix2Pix generator architecture.

1.1. Summary of Our Contributions

We introduce for the first time a convolutional capsule network in the GAN framework. We propose a convolutional capsule architecture—CapsPix2Pix—for conditional image generation, that utilises an input label and a latent vector (see Fig. 1). We show that CapsPix2Pix quantitatively performs as well as the state-of-the-art `pix2pix` (Isola et al., 2017) for generating realistic images with our dataset, while drastically reducing the number of network parameters: 7× smaller than `pix2pix` (7.9M vs. 50M parameters). We show that capsules

capture qualitatively different features for the relevant structures in our dataset, and can create varied synthetic images from the same labels (Fig A6, & Fig. 2). We also explore the features learned by convolutional capsules. Through this we find that they group similar features in the same capsule. Finally, we present a new dataset of cortical neurons and segmentation labels collected using two-photon microscopy in the mouse cortex. The dataset is available at https://doi.org/10.5281/zenodo.2559237, and the code for our method can be found at https://github.com/CherBass/CapsPix2Pix.

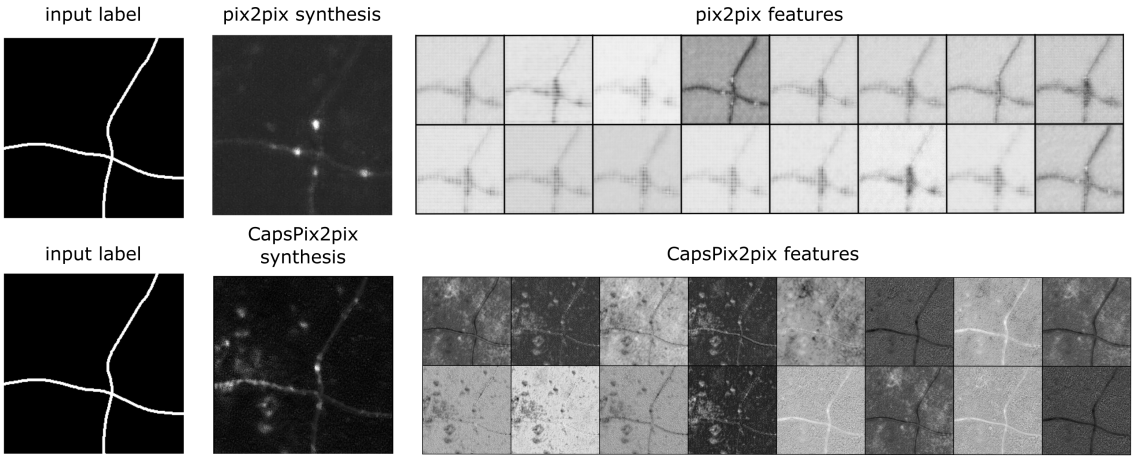

Figure 2: Comparison of the features of `pix2pix` (16/64 last layer activations— see Fig. A7 for all activations) and CapsPix2Pix (16/16 last layer activations) from the same geometric description of an axon.

2. Background and Related Work

Recent years have seen the introduction of several methods that are capable of conditional image synthesis. Such methods may take as input segmentation labels (Isola et al., 2017), object bounding boxes (Reed et al., 2016b) or even text (Reed et al., 2016a), and produce realistic images conditioned on this information. For biomedical data, the ability to synthesise new samples would allow us to apply powerful supervised learning methods to datasets with few labels—effectively, semi-supervised learning by synthesising new labelled samples. Accordingly, some prior work has used such methods to generate synthetic biomedical datasets (Hinterstoisser et al., 2017; Alzantot et al., 2017; Frid-Adar et al., 2018a,b; Sixt et al., 2018; Korkinof et al., 2018; Baur et al., 2018).

In this work we build upon `pix2pix` (Isola et al., 2017), a conditional generative adversarial network (cGAN) (Goodfellow et al., 2014; Mirza and Osindero, 2014) that is capable of synthesising images conditional on segmentation labels. We augment `pix2pix` with convolutional capsules (Sabour et al., 2017; LaLonde and Bagci, 2018), which are proposed to be better able to capture relationships between features than standard convolutional NNs (CNNs).

2.1. Generative Adversarial Networks

GANs are a type of implicit generative model that map latent vectors $z \sim P_z(z)$ to samples y, via a generator model $G : z \to y$ (Goodfellow et al., 2014). They can be extended to the conditional setting (Mirza and Osindero, 2014), where the mapping is from z and an additional input label x, such that $G : \{x, z\} \to y$. For the case of our biomedical dataset, x represents the geometry of the structures to be synthesised, z captures additional properties of the data distribution, such as imaging noise and variations in axon intensities, and y is a sample resembling a cortical axon image.

GAN training also involves a discriminator model $D : \{x, y\} \to [0, 1]$, where D is shown both real and synthetic image-label pairs and is trained to distinguish between them via a binary classification task. G is then trained to generate samples that fool the discriminator. Training is formulated as a two-player minimax game, where the objective is to find a Nash equilibrium for both models:

$$\begin{aligned}\min_G \max_D L_{cGAN}(G, D) =& \mathbb{E}_{x,y\sim P_{\text{data}}(x,y)}\big[\log D(x, y)\big] + \\ & \mathbb{E}_{x\sim P_{\text{data}}(x), z\sim P_z(z)}\big[\log(1 - D(x, G(x, z)))\big]\end{aligned} \tag{1}$$

For more information on (conditional and unconditional) GANs we refer readers to Goodfellow (2016); Creswell et al. (2018).

The `pix2pix` network (Isola et al., 2017) is a fully-convolutional cGAN based on a U-net-style encoder-decoder architecture (Ronneberger et al., 2015). In order to encourage the final output to better adhere to the structure of the label, they also minimise an L_1 loss between synthesised and real images with the corresponding label:

$$\min_G L_1(G) = \mathbb{E}_{x,y\sim P_{\text{data}}(x,y), z\sim P_z(z)}\big[\, \| \, y - G(x, z) \, \|_1 \,\big] \tag{2}$$

The final objective, which we also use for CapsPix2Pix, is the value function $V(G, D)$:

$$\min_G \max_D V(G, D) = L_{cGAN}(G, D) + \lambda L_1(G) \tag{3}$$

where $\lambda = 0.1$ for `pix2pix`, and $\lambda = 1$ for CapsPix2Pix (see appendix section 7.3 for results comparing $\lambda \in \{0.1, 1\}$ for CapsPix2Pix). The original formulation of `pix2pix` also forgoes using $z \sim P_z(z)$ to generate a distribution of images, and instead uses dropout to stochastically generate outputs, as they found that their generator's outputs were largely independent of z. We instead retain the approach of inputting random latent vectors to drive CapsPix2Pix, as we found that it was capable of learning meaningful manifold in latent space (Fig. 5).

2.2. Capsule Networks

Capsule networks were first introduced by Sabour et al. (2017), and were made to better encode spatial relationships between features than standard CNNs. As opposed to standard DNNs, where scalar values represent a feature, capsules output vectors, where the orientation of the vector represents properties (e.g., pose, texture, etc.), and the magnitude represents the probability of the feature being present. The second key component of capsule networks is "dynamic routing", an iterative algorithm in which outputs of capsules are

routed to capsules in the layer above based on how well their predictions agree. Our version of the dynamic routing algorithm is detailed in Algorithm 1.

However, traditional capsules use a high-dimensional transformation matrix, and thus were only applied to small images. The size of the weight matrix is also fixed based on the input size, and so the same network cannot be applied to different sized images. LaLonde and Bagci (2018) solved this issue by introducing convolutional capsules, and successfully applied them to a segmentation task with large images (512×512 pixels). As in standard CNNs, convolutional capsules benefit from a smaller memory usage and faster computation. By virtue of having spatial filters, they are also more amenable to analysis by examining layer activations.

We note that there have been two prior works (Jaiswal et al., 2018; Upadhyay and Schrater, 2018) that have combined capsules with GANs by changing the discriminator model to use (non-convolutional) capsules. They therefore were unable to demonstrate the benefits of using capsules in the generator model, and did not demonstrate an application to images beyond (64×64 pixels). In initial experiments (see appendix section 7.3), we found that standard convolutional discriminators (Radford et al., 2015) qualitatively performed better than convolutional capsule discriminators, and so opted to use the former.

3. Methods

3.1. Convolutional Capsules & Dynamic Routing

Our generator utilises convolutional capsules (LaLonde and Bagci, 2018), where the dynamic routing algorithm is instead preceded by a convolution instead of a weight matrix transformation. This allows the network to learn relationships between all capsules in a given layer ("child" capsules) and all capsules in the layer above ("parent" capsules).

Convolutional capsule layers take inputs a, of size $[B, I, CH_I, W_I, H_I]$, and output $\hat{u}$, of size $[B, I, W_J, H_J, J, CH_J]$, where B = batch size, I = number of input capsules, CH_I = number of input channels, W = width, H = height, J = number of output capsules and CH_J = number of output channels. In the convolution step, the kernels $K_W \times K_H$ are shared between the input capsules in order to reduce the number of parameters, such that the total number of parameters per convolution is $CH_I \times K_W \times K_H \times CH_J \times J$. For each layer of the network, the output of the convolution $\hat{u}$—the activations of the child capsules—are routed to all the parent capsules. We use i and j to denote indices for child and parent capsules, respectively.

The routing process involves taking the dot product of $\hat{u}_{ij}$ with vector c_{ij} over the input capsules to give s_j. c_{ij} is computed using the softmax function:

$$c_{ij} = \frac{\exp(b_{ij})}{\sum_i \exp(b_{ij})} \tag{4}$$

where b_{ij} is updated in every routing iteration. For convolutional capsules, b has dimensions $[I, W_J, H_J, J]$. s_j is then passed through the nonlinear "squash" function:

$$v_j = \frac{\|s_j\|_2^2}{1 + \|s_j\|_2^2} \cdot \frac{s_j}{\|s_j\|_2} \tag{5}$$

The purpose of this nonlinear function is to normalise the output between $[0, 1]$, and to ensure that the vector direction is retained, which is important in the b_{ij} update step. The update step of b_{ij} is taking the dot product of v_j with $\hat{u}_{ij}$, which is a key element of the dynamic routing algorithm. The dot product essentially looks at the similarity between the input and output capsules, and updates b_{ij} accordingly (similar features increase the value of b, and dissimilar features reduce it).

The entire procedure for convolutional capsules followed by dynamic routing is illustrated in Algorithm 1, where we use $*$ to denote either convolution or transpose convolution.

Algorithm 1: Convolutional Capsules + Dynamic Routing

Input: a, capsules in layer l; l, layer; r, iterations; bias; weight
Output: v_j, capsules in layer $(l+1)$
$\hat{u}_{I\times J\times CH_J} \leftarrow bias_{J\times CH_J} + \sum_{n=0}^{CH_I} weight_{J\times CH_J,n} * a_n$
for all capsules i in layer l and capsules j in layer $(l+1)$: $b_{ij} \leftarrow 0$
for *1* **to** r **do**
  for all capsules i in layer l and capsule j in layer $(l+1)$: $c_{ij} \leftarrow softmax(b_{ij})$ ▷ Eq. 4
  for all capsules j in layer $(l+1)$: $s_j \leftarrow \sum_i c_{ij}\hat{u}_{ij}$
  for all capsules j in layer $(l+1)$: $v_j \leftarrow squash(s_j)$ ▷ Eq. 5
  for all capsules i in layer l and capsule j in layer $(l+1)$: $b_{ij} \leftarrow b_{ij} + \hat{u}_{ij} \cdot v_j$
end

3.2. Convolutional Capsule Generator

Our generator architecture is based on the U-net-style encoder-decoder network with skip connections (Fig. 1) (Ronneberger et al., 2015), similarly to `pix2pix` (Isola et al., 2017) and SegCaps (LaLonde and Bagci, 2018). This architecture is designed to output an image that is aligned with the structure of the input, and to encode both low and high level features by using skip connections. We later demonstrate that capsules learn highly relevant features at the last layer, while having far fewer parameters as compared to the original `pix2pix`.

In order to generate a distribution of synthetic images, we additionally augment our network with a 100D latent vector (Fig. 1) $z \sim \mathcal{N}(0, \mathbb{I})$. This vector is passed through a fully-connected layer of size 100×256^2, with the output reshaped to be the same spatial dimensions as the input image (256×256). This is then concatenated with the input segmentation label as an additional channel, so that the input into the first convolutional layer is of size $[B, 2, 256, 256]$. By fixing the input segmentation label and sampling different latent vectors, the network is able to produce varied images with the same geometry as represented by the label.

3.3. Conditional DCGAN Discriminator

Our discriminator is mainly based on the deep convolutional GAN (DCGAN) architecture (Radford et al., 2015). As per the original, the network is built with standard 2D convolutions followed by batch normalisation (Ioffe and Szegedy, 2015) and leaky ReLU nonlinearities (Xu et al., 2015); we also add a final fully-connected layer of size 169×1 to

compensate for the larger input size. To make it conditional, D receives as an input the image and label concatenated, giving an input size $[B, 2, 256, 256]$.

4. Datasets

We used both a real microscopy dataset and a physics-based dataset in our experiments (Fig 3). These datasets comprise of both labels and images. In addition, we also describe several methods that can take labels (from either dataset) as input, and generate new images.

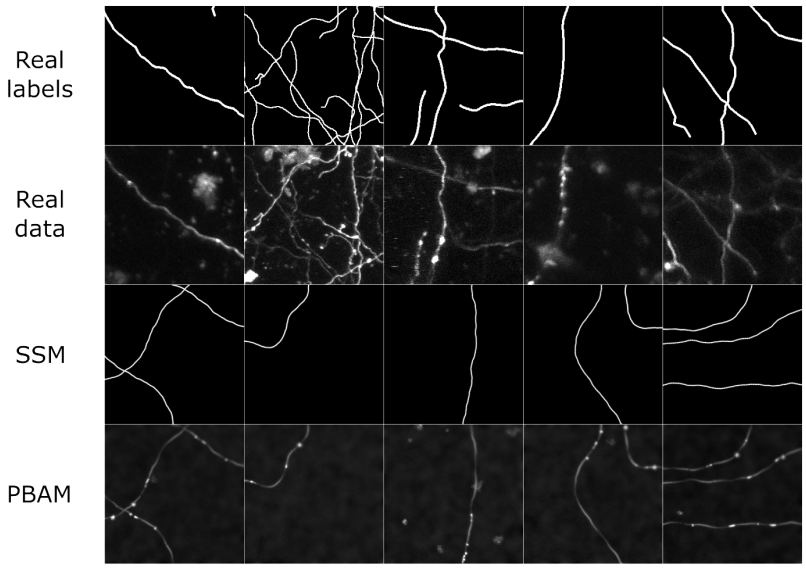

Figure 3: Examples of labels and images from the real and physics-based synthetic datasets. Abbreviations: SSM = statistical shape model; PBAM = physics-based imaging appearance model.

4.1. Synthetic datasets

The synthetic dataset creates labels from a statistical shape model (SSM). The algorithm starts with an empty matrix that is progressively filled with points constructed by an angle-constrained random walk. These permit an infinite number of variations of axon-like geometry to be specified. The skeleton of the axon-like structures is obtained by fitting spline curves through these points (Wen and Chklovskii, 2008). Labels are then defined by keeping all pixels within a short distance from the curves.

PBAM-SSM: The corresponding images can be synthesised by using the spline-specified synthetic geometry to drive a simple physics-based imaging appearance model (PBAM). This model first builds a distance map between the axon centreline and the rest of the

image. Distances are converted into intensity values by assuming a Gaussian axon profile or similar centre-line dependent, variable width intensity profile. The final image is obtained by introducing intensity variations along the axons, optical blur, white and coloured noise sources. Synaptic protrusions are also added onto the axons. The combined geometry/appearance model is used as a baseline generative model for microscopy images.

CapsPix2Pix-SSM & pix2pix-SSM: The statistical shape model representing axons can be used to drive different geometrical realisations for the appearance-based generative models based on either the CapsPix2Pix or `pix2pix` networks.

4.2. Microscopy dataset

We combined data from two published sources (Bass et al., 2017; Canty et al., 2018) to get 152 512×512 2D images (produced from a max projection over 3D image stacks), and manually produced the corresponding labels; 20 of the images are held out for testing. These images were collected using in-vivo two-photon microscopy from the mouse somatosensory cortex. Examples of the labels and images in this dataset are shown in Fig. 3. The labels are binary segmentation maps of the axons. The full dataset is available at https://doi.org/10.5281/zenodo.2559237.

4.3. Training of networks

The conditional generative models `pix2pix` and CapsPix2Pix are trained on labels and images from the microscopy dataset, which is augmented with random crops of size 256×256, which we denote as "cropped real" (CR). The segmentation models (U-nets) are trained using data synthesised from labels obtained from the SSM, and labels from the microscopy dataset, augmented with random flips, rotations, zooming and crops of size 64×64, which we denote as "augmented real" (AR). Examples of synthetic images from different models compared to real data are shown in Fig. 4. We use 26,400 images in all our experiments for training the segmentation and conditional generative models.

5. Experiments and Results

5.1. Evaluation of Generative Models

Evaluating the quality of generative models is known to be a difficult problem (Theis et al., 2016). Several quantitative methods, such as Parzen window log-likelihood estimates (Breuleux et al., 2010) and the Inception score (Salimans et al., 2016) have been proposed, but these may not always be appropriate. We provide a further discussion in appendix section 7.2. In the case of conditional image synthesis, we could use quantitative metrics such as mean squared error or structural similarity index to compare against "ground truth" images. Unfortunately, these do not give a meaningful evaluation of the quality of the synthesis, since if the structure is similar, but the quality or data distribution is bad, the scores might still be high. A more general way of evaluating synthetic image quality is to allow human observers to discriminate between "real" and "fake" images. However, as our dataset contains biomedical images, subjects would need to be trained on the data first before performing the task. We instead choose to quantitatively evaluate our method by testing how its synthesised images affect the performance in downstream

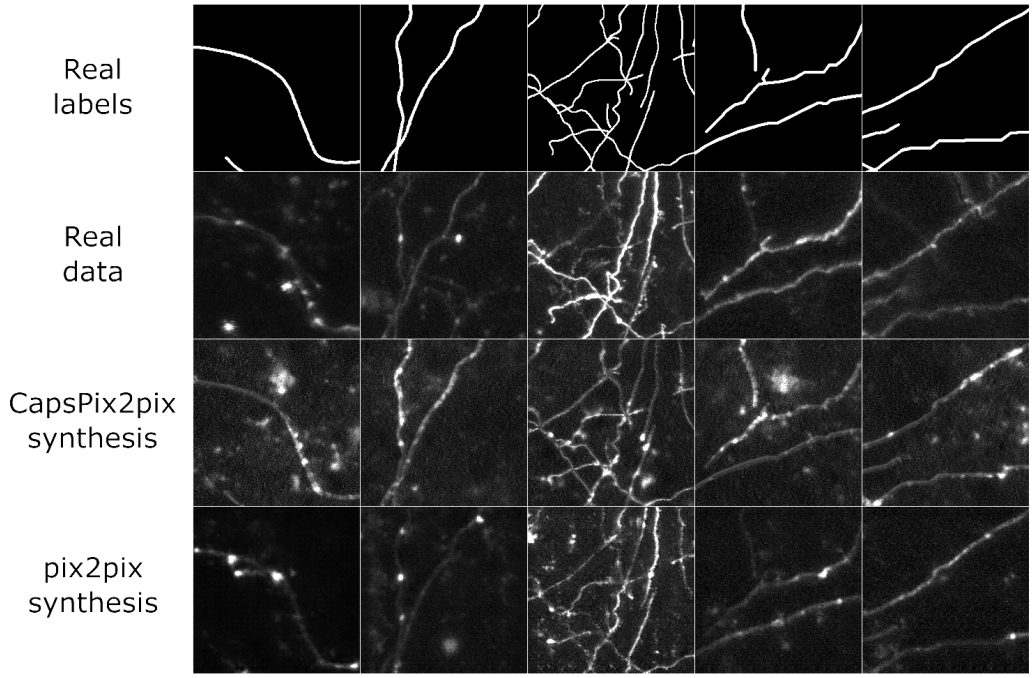

Figure 4: Examples of synthetic images compared to images from the real dataset. All images have a resolution of 256×256 pixels.

tasks of interest—in this case, segmentation. This task-based evaluation has previously been used for evaluating generative models in, for example, skin lesion datasets (Frid-Adar et al., 2018a,b). Qualitatively, it is possible to examine the latent space to see how well the network has learned high level information about the data distribution, and the network's activations to see what features have been learned.

5.2. Quantitative Analysis

As the end goal of our work on generative models is to improve the analysis of biomedical datasets, we chose to quantitatively evaluate our model by using synthesised images in a segmentation task. We trained separate U-nets from scratch, on different datasets (real, and synthetic images from different models), and repeat each experiment 10 times to get means and standard deviations per model. Testing was done on the same 400 crops (64×64) of held-out test dataset of 20 images (512×512). Performance was measured using the Dice score, receiver operator characteristic (ROC) area under the curve (AUC), and precision-recall (PR) AUC metrics (Table. 1). We compared our model, CapsPix2Pix, to the state-of-the-art `pix2pix` (Isola et al., 2017). See ROC and test plots for selected U-nets (trained on different datasets) in Figs. A3-A4.

Table 1: **Segmentation results.** The same U-net architecture was trained on different datasets, and evaluated on the test set of 400 crops of size 64×64 from the original dataset (20 test images of size 512×512). The same number of images were used in training in all experiments, 26,400 of size 64×64. Hyphenated names refer to image synthesis model, followed by the label source. † refers to the usage of only 1,320 unique labels, but generating 20 images per label. We highlight in bold the best scores in each section separated by a horizontal line. We repeat each experiment 10 times, and report averages and standard deviations across those experiments in this table. Abbreviations: PBAM = physics-based imaging appearance model; SSM = (synthetic labels of the) statistical shape model; AR = augmented real (labels).

Images	Labels	Pretrained	Dice	ROC AUC	PR AUC
PBAM	SSM	No	0.6094±0.008	0.9560±0.005	0.6157±0.01
`pix2pix`	AR	No	0.6592±0.003	0.9670±0.0004	0.6834±0.003
`pix2pix`	SSM	No	0.6353±0.003	0.9631±0.001	0.6635±0.006
CapsPix2Pix	AR	No	0.6407±0.004	0.9608±0.002	0.6572±0.005
CapsPix2Pix	SSM	No	0.6528±0.002	0.9637±0.001	0.6691±0.0002
Real data	AR	No	**0.6827±0.001**	**0.9725±0.0002**	**0.7149±0.001**
`pix2pix`	AR†	No	0.6438±0.003	**0.9652±0.0009**	0.6676±0.004
`pix2pix`	SSM†	No	0.6393±0.004	0.9635±0.001	0.6657±0.004
CapsPix2Pix	AR†	No	0.6423±0.007	0.9638±0.001	0.6621±0.01
CapsPix2Pix	SSM†	No	**0.6540±0.007**	0.9620±0.004	**0.6689±0.006**
Real data	AR	`pix2pix`-AR	0.6856±0.0005	0.9731±0.0002	0.7170±0.0009
Real data	AR	`pix2pix`-SSM	0.6830±0.0005	0.9724±0.0002	0.7145±0.0007
Real data	AR	CapsPix2Pix-AR	**0.6870±0.0005**	**0.9734±0.0001**	**0.7190±0.0006**
Real data	AR	CapsPix2Pix-SSM	0.6855±0.0005	0.9730±0.0001	0.7175±0.0006

Training U-net from scratch on real or synthetic datasets: When comparing the Dice score per experiment (Table. A1), we found that the CapsPix2Pix-SSM model performs better than the `pix2pix`-SSM model ($p < 0.0001$, t-test), and better than the PBAM-SSM ($p < 0.0001$, t-test). However, the `pix2pix`-AR model performs better than CapsPix2Pix-AR ($p < 0.0001$, t-test). We conclude that, overall, training U-net *from scratch* on CapsPix2Pix or `pix2pix` synthetic data leads to comparable results, since they each perform better in one case when using either SSM or AR as input labels. Overall, using the real data with augmentation leads to better segmentation results than training only on the synthetic images ($p < 0.0001$, t-test).

Pre-training U-net on synthetic datasets: To test whether using the synthetic datasets could improve upon only training with real images, we pretrain our U-nets on synthetic images, and fine-tuned using the real data. Doing so significantly improves the Dice score (compared to just training on real data) when pretraining on `pix2pix`-AR data ($p < 0.0001$, t-test), but not on `pix2pix`-SSM ($p = 0.36$, t-test). In contrast, we achieve a significant improvement on both CapsPix2Pix-AR and CapsPix2Pix-SSM data (AR & SSM; $p < 0.0001$, t-test). Overall, we reach the *best results* when pretraining on CapsPix2Pix-AR

(0.6870 ± 0.00005), and we find that it is significantly better than the pretrained model on `pix2pix`-AR data ($p = 0.00003$, t-test).

Training U-net on synthetic datasets with reduced unique labels: We performed an additional experiment to quantitatively demonstrate that our model can synthesise more diverse data, given the same input label. We trained U-nets with a reduced number of SSM or AR labels (1,320 unique labels), but several synthetic images per label (20 images) for a total of 26,400 (i.e. same number of images as in other experiments), for both CapsPix2Pix and `pix2pix`. We found that while training on CapsPix2Pix-SSM† data (where † refers to a model trained with reduced labels) leads to good performance, training on `pix2pix`-SSM† data leads to overfitting, and performance is reduced to ~ 0.61 by the end of training (bottom of Fig. A4). In addition, when comparing performance of the best models (i.e. before overfitting, Table 1), we found that CapsPix2Pix-SSM† is significantly better than `pix2pix`-SSM† and `pix2pix`-AR† (p=0.000086, and p=0.0017, t-test). This strongly indicates that our model is better able to synthesise diverse images. See Fig. A6 for randomly picked examples of synthetic images from the same input labels.

5.3. Qualitative Analysis

We investigated how well our model captured the data distribution of the axon and noise using linear interpolation between 2 randomly sampled z vectors (Fig. 5). The synthetic images display a large variation in noise and axon intensity (including synapses on different locations on the axon). We also display the features (activations) of the last capsule layer, demonstrating the variety of features learned by the network, even with a reduced number of parameters. In addition, we found that individual capsules appear to group similar features (Fig. A5). This could be key in learning a balanced data distribution within and across classes, as different capsules could represent different classes, and the features within each capsule could encode the variability within that class.

We also compared the 16 features (the last layer activations) produced from the same label between `pix2pix` and CapsPix2Pix (Fig. 2). While `pix2pix` also learns different features, many of them appear to be redundant and do not capture the noise very well (See Fig. A7 for all `pix2pix` features at the last layer). CapsPix2Pix learns both features of the axon and noise (high intensity, and general noise) quite well.

Finally, we synthesised different images for the same label from each network (Fig. A6), and found that while CapsPix2Pix can generate a large variation of images from one label, `pix2pix` is only able to make minor alterations to the images.

6. Conclusion and Discussion

In this paper we introduced CapsPix2Pix, a novel method which utilises convolutional capsule networks in the cGAN framework for synthesising images conditional on segmentation labels.

We quantitatively validate our method by training U-nets on synthesised data from CapsPix2Pix and the state-of-the-art `pix2pix` model, which leads to comparable performance in a segmentation task (see section 5.2), while using far fewer parameters (see appendix section 7.1). More relevant to our end goal, we show that if the U-net is pretrained with synthesised data from CapsPix2Pix, it increases the performance in the segmentation task.

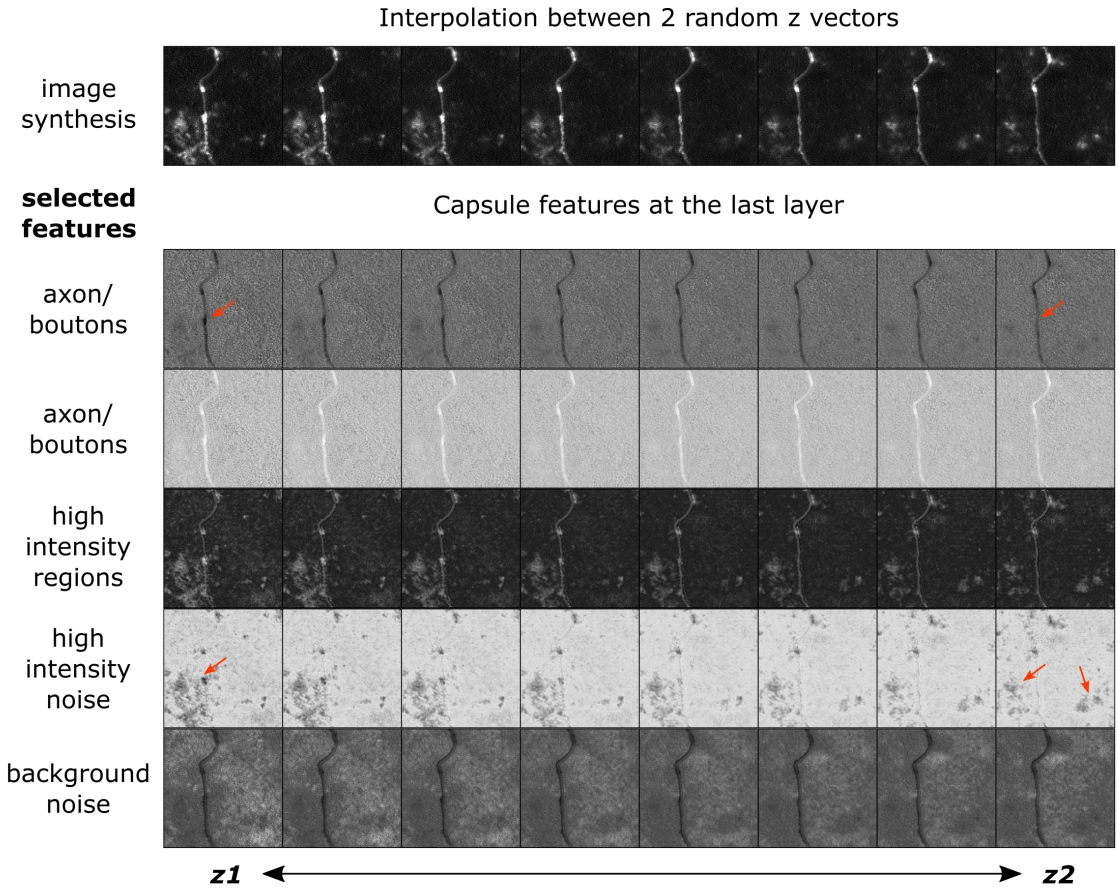

Figure 5: Linear interpolation between two z vectors for CapsPix2Pix. We show 5 selected activations/ features from our 16D capsule at the last layer. The red arrows in the axon/ boutons row, point to an example of how a bouton appears and disappears in the feature space, when interpolating the z vectors. The red arrows in the high intensity noise row, point to changing high intensity noise.

We attempt to compare the data distributions of the real and synthetic images via kernel density estimation, but conclude that this method is invalid in our case (see discussion in appendix section 7.2).

Qualitatively, we show that CapsPix2Pix is able to synthesise considerably different images from the same images, while `pix2pix` does not (see section 5.3, Fig. A6). We quantified whether being able to synthesise different images from the same label would impact the results when training U-net. We synthesised images from a reduced number of unique labels with 20 images per label, and showed that segmentation performance was better when CapsPix2Pix data was used (see section 5.2, bottom of Fig. A4). We also show that CapsPix2Pix learns relevant features for our dataset, and appears to capture the distribution of the noise and neuron classes well (Figs. 2, 5, A5), while `pix2pix` mainly

just captures the axon class (Figs. 2, A7). Because of this property, we hypothesise that CapsPix2Pix could generalise well to other or more complicated datasets with multiple classes, and possibly even improve on the problem of mode collapse. For example, it could apply well to other biomedical datasets, where class imbalance is a common problem, and the dataset could benefit from a generator that captures a balanced distribution both within and across classes. In particular, our model's ability to synthesise diverse images could be of greater importance when dealing with datasets with a small amount of labelled data—another common problem in biomedical datasets.

Acknowledgments

This work was supported by the Engineering and Physical Sciences Research Council [grant number EP/L016737/1].

References

Moustafa Alzantot, Supriyo Chakraborty, and Mani Srivastava. Sensegen: A deep learning architecture for synthetic sensor data generation. In *Pervasive Computing and Communications Workshops (PerCom Workshops), 2017 IEEE International Conference on*, pages 188–193. IEEE, 2017.

Cher Bass, Pyry Helkkula, Vincenzo De Paola, Claudia Clopath, and Anil Anthony Bharath. Detection of axonal synapses in 3d two-photon images. *PloS one*, 12(9):e0183309, 2017.

Christoph Baur, Shadi Albarqouni, and Nassir Navab. Melanogans: High resolution skin lesion synthesis with gans. *arXiv preprint arXiv:1804.04338*, 2018.

Olivier Breuleux, Yoshua Bengio, and Pascal Vincent. Unlearning for better mixing. *Universite de Montreal/DIRO*, 2010.

Alison J Canty, Johanna S Jackson, Lieven Huang, Antonio Trabalza, Cher Bass, Graham Little, and Vincenzo De Paola. Single-axon-resolution intravital imaging reveals a rapid onset form of wallerian degeneration in the adult neocortex. *bioRxiv*, page 391425, 2018.

Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE Signal Processing Magazine*, 35(1):53–65, 2018.

Maayan Frid-Adar, Idit Diamant, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. Gan-based synthetic medical image augmentation for increased cnn performance in liver lesion classification. *arXiv preprint arXiv:1803.01229*, 2018a.

Maayan Frid-Adar, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. Synthetic data augmentation using gan for improved liver lesion classification. In *Biomedical Imaging (ISBI 2018), 2018 IEEE 15th International Symposium on*, pages 289–293. IEEE, 2018b.

Ian Goodfellow. Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

Stefan Hinterstoisser, Vincent Lepetit, Paul Wohlhart, and Kurt Konolige. On pre-trained image features and synthetic images for deep learning. *arXiv preprint arXiv:1710.10710*, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint*, 2017.

Ayush Jaiswal, Wael AbdAlmageed, and Premkumar Natarajan. Capsulegan: Generative adversarial capsule network. *arXiv preprint arXiv:1802.06167*, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Dimitrios Korkinof, Tobias Rijken, Michael O'Neill, Joseph Yearsley, Hugh Harvey, and Ben Glocker. High-resolution mammogram synthesis using progressive generative adversarial networks. *arXiv preprint arXiv:1807.03401*, 2018.

Rodney LaLonde and Ulas Bagci. Capsules for object segmentation. *arXiv preprint arXiv:1804.04241*, 2018.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS Autodiff Workshop*, 2017.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning-Volume 48*, pages 1060–1069. JMLR. org, 2016a.

Scott E Reed, Zeynep Akata, Santosh Mohan, Samuel Tenka, Bernt Schiele, and Honglak Lee. Learning what and where to draw. In *Advances in Neural Information Processing Systems*, pages 217–225, 2016b.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pages 3856–3866, 2017.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242, 2016.

Leon Sixt, Benjamin Wild, and Tim Landgraf. Rendergan: Generating realistic labeled data. *Frontiers in Robotics and AI*, 5:66, 2018.

L Theis, A van den Oord, and M Bethge. A note on the evaluation of generative models. In *International Conference on Learning Representations (ICLR 2016)*, pages 1–10, 2016.

Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.

Yash Upadhyay and Paul Schrater. Generative adversarial network architectures for image synthesis using capsule networks. *arXiv preprint arXiv:1806.03796*, 2018.

Quan Wen and Dmitri B. Chklovskii. Costbenefit analysis of neuronal morphology. *Journal of Neurophysiology*, 2008.

Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

7. Appendix

7.1. Computational Comparison

Here we shown a comparison of the computational footprint of CapsPix2Pix and `pix2pix`.

Weights and activations: We compared the number of trainable parameters for the generator of CapsPix2Pix and `pix2pix`. CapsPix2Pix has 7.9M parameters while `pix2pix` has in total 54M parameters, i.e. CapsPix2Pix is $\sim \times 7$ smaller. The size of the activations in the network is comparable between the two networks—with CapsPix2Pix having a total size of 18M and `pix2pix` having a total size of 16M.

Run-time in training and inference: We found that during training, running 1 training epoch for a batch size of 1, for both generator and discriminator, takes CapsPix2Pix 0.27 seconds and `pix2pix` 0.055 seconds, i.e. CapsPix2Pix is $\sim \times 5$ slower than `pix2pix`. During inference, isolating 1 run through the generator for 1 image, we found that it takes CapsPix2Pix 0.058 seconds, and `pix2pix` 0.00445 seconds, i.e. CapsPix2Pix is $\sim \times 13$ slower than `pix2pix` when comparing the generator only. These run-time experiments were performed on the same PC with a GeForce GTX 1080Ti GPU.

7.2. Quantitative Metrics for Evaluating Generative Models

Ideally, we would be able to quantify how well samples from a generative model match samples from the real data distribution. This is one motivation for using Parzen window estimates (Breuleux et al., 2010)—in which a nonparameteric kernel density estimator is fit to real data, giving the ability to evaluate log-likelihoods for model samples against the density estimator. We performed this evaluation, fitting 100 Gaussian kernels, with bandwidths ranging from 0.1 to 0.4, to 10000 samples of real data (32×32), and evaluated the log-likelihoods of 10 draws of 1000 samples of synthetic data (`pix2pix`, CapsPix2Pix). As a control, we also evaluated 10 draws of 1000 samples from left-out real data. The results can be seen in Fig. A1: while CapsPix2Pix has a higher log-likelihood than `pix2pix` across all bandwidths, real data has the lowest log-likelihoods. This has been shown to occur before, and unfortunately invalidates its use as a quantitative metric (Theis et al., 2015).

Another metric—the Inception score—is based on the Kullback-Leibler divergence between the conditional label distribution, $p(y|x)$, and the marginal label distribution, $p(y)$, where the distributions are evaluated using a pretrained discriminative network (originally Inception-v3 trained on ImageNet data) (Salimans et al., 2016). A high score corresponds to generating meaningful objects (the entropy of $p(y|x)$ is low) and a wide range of classes (the entropy of $p(y)$ is high). While this metric may make sense for image datasets with a large number of classes with a balanced set of samples, it breaks down for a small number of classes with imbalanced data—as is the case for our axon dataset (Bass et al., 2017). In particular, generated samples with a low amount of noise could potentially score higher, although this is unrealistic.

The Frchet inception distance compares the activations in a pretrained discriminative network (originally Inception-v3) of both real and generated samples (Heusel et al., 2017). This comparison is similar to Parzen window estimation, but occurs in a relevant feature space rather than on raw features. The feature space of the pretrained network is important. The image statistics of generic, real images are vastly different to those of medical images,

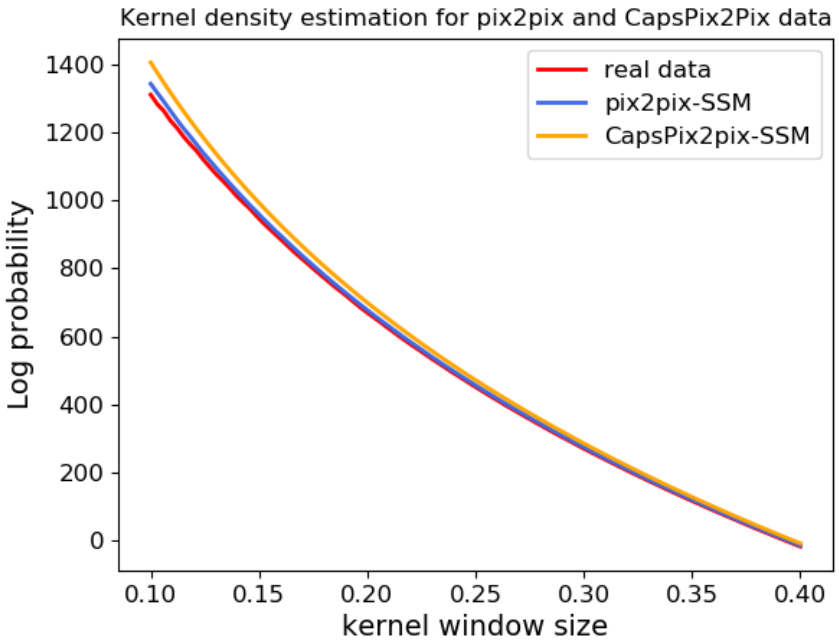

Figure A1: Kernel density estimation plot, comparing the average (of 10 experiments) log-likelihoods of samples from `pix2pix`, CapsPix2Pix and held-out real data. As the estimated log-likelihood of the real data is consistently below generated data, this invalidates its use as a quantitative metric. Abbreviation: SSM = (synthetic labels of the) statistical shape model.

requiring a network pretrained on medical imaging data. However, the previously described problem with noise re-occurs, as we typically train discriminative networks to be robust to noise in the input images.

7.3. Additional Training Experiments For CapsPix2Pix

We performed some initial experiments to explore possible configurations for CapsPix2Pix. We tested whether using the same L1 lambda ($\lambda = 0.1$) as in `pix2pix` (Isola et al., 2017) would work well for CapsPix2Pix. We found that while the synthetic images were reasonable, the images were too noisy at times, and might be unrealistic in some cases (Fig. A2). Also, we found that during training, the quality of synthetic images oscillated a lot more than when using $\lambda = 1$. We also experimented with using different discriminator networks, including a network based on the traditional capsules (Sabour et al., 2017; Jaiswal et al., 2018; Upadhyay and Schrater, 2018), and one based on convolutional capsules (this network was similar in structure to a DCGAN, i.e., using batch normalisation, leaky ReLUs, the same number of convolutional layers, etc). We found that the traditional capsule discriminator did not work well at training the generator network, and that the convolutional capsule discriminator led to reasonable synthetic image generation, but was not as good as

when using a DCGAN discriminator (Fig. A2). In addition, both capsule-based discriminator networks increased training time. Lastly, we experimented with different ways to insert noise into the network, including broadcasting noise, adding it to the bottleneck, or using dropout in inference as in `pix2pix`, but these initial experiments were not promising.

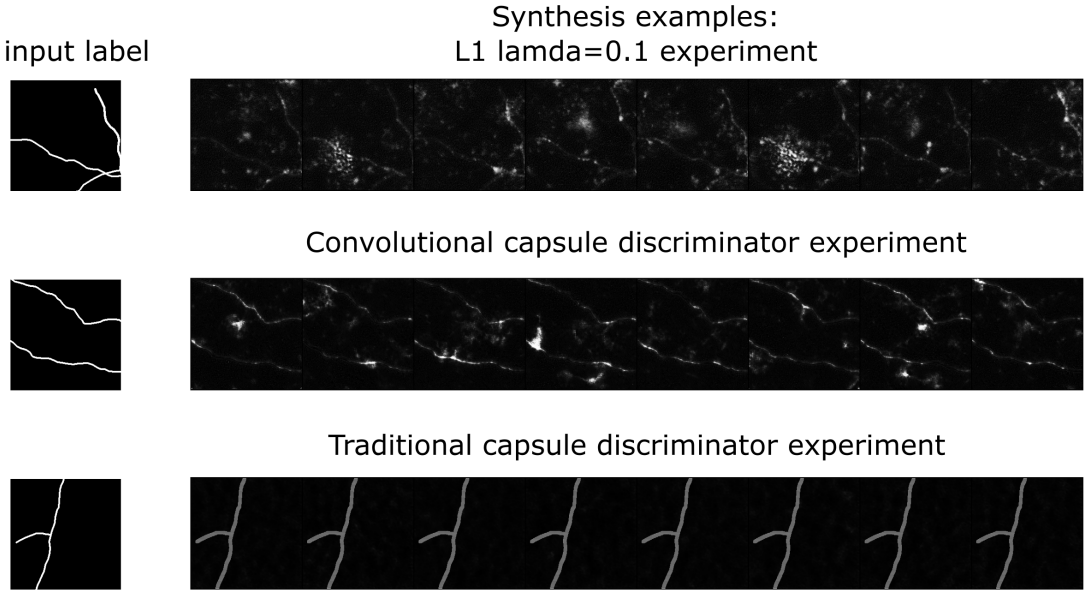

Figure A2: Additional training experiments.

7.4. CapsPix2Pix Training Details

To optimise our network, we follow the approach laid out by Goodfellow et al. (2014). We train G to maximize $\log D(x, G(x, z))$, instead of minimizing $\log(1 - D(x, G(x, z)))$. In addition, we use Adam (Kingma and Ba, 2014) as our optimiser, with learning rate $= 0.0002$, $\beta_1 = 0.5, \beta_2 = 0.999$; we linearly decay learning rate to 0 by the end of training. To prevent D from becoming overconfident we use two-sided label smoothing (Salimans et al., 2016), with positive labels set to 0.9 and negative labels set to 0.1 when updating D. We also train G with a dropout probability of 0.5, which we do not apply during inference. Due to memory constraints we use small minibatch sizes (1-4) and hence do not use batch normalisation to train G.[1] We train D with batch normalisation, as in the original DCGAN. Unlike `pix2pix`, we input a latent vector z, so we do not need to add additional stochasticity to our generator during inference.

We trained the networks on the CR dataset (see section 4). Examples of the synthetic images are shown in Fig. 4. The same dataset was used to train `pix2pix` for comparison. We trained `pix2pix` as described in the original work (Isola et al., 2017). All experiments were implemented using PyTorch (Paszke et al., 2017).

1. We found that using batch normalisation is effective when training on smaller images [64×64], and using higher batch size is possible.

7.5. U-net Segmentation Training Details

For each experiment we trained a standard U-net (with same padding so that the output has the same image size). These were trained on 26,400 (64×64) crops from various datasets, and were tested on the same 400 crops from the test dataset (20 images, 512×512). During training we used a dropout probability of 0.5, a batch size of 32, and Adam with learning rate $= 0.00001$, $\beta_1 = 0.5, \beta_2 = 0.999$. We keep aside 20% of the training data for validation.

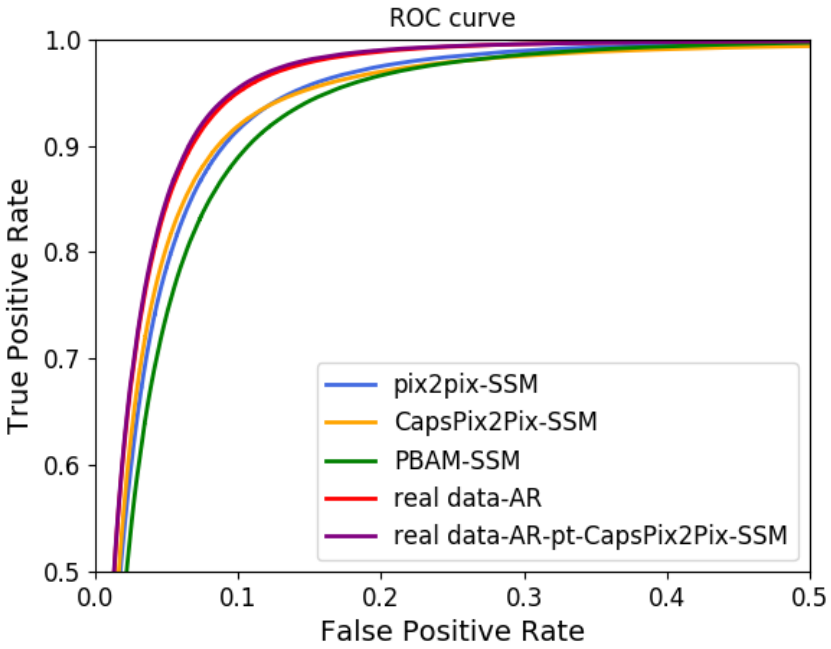

Figure A3: Comparison of U-net test results using ROC curves for different datasets. Each ROC curve is a concatenation of all false positive rates and true positive rates per dataset ($\times 10$ experiments). Abbreviations: PBAM = physics-based imaging appearance model; SSM = (synthetic labels of the) statistical shape model; AR = augmented real (labels); pt = pretrained network.

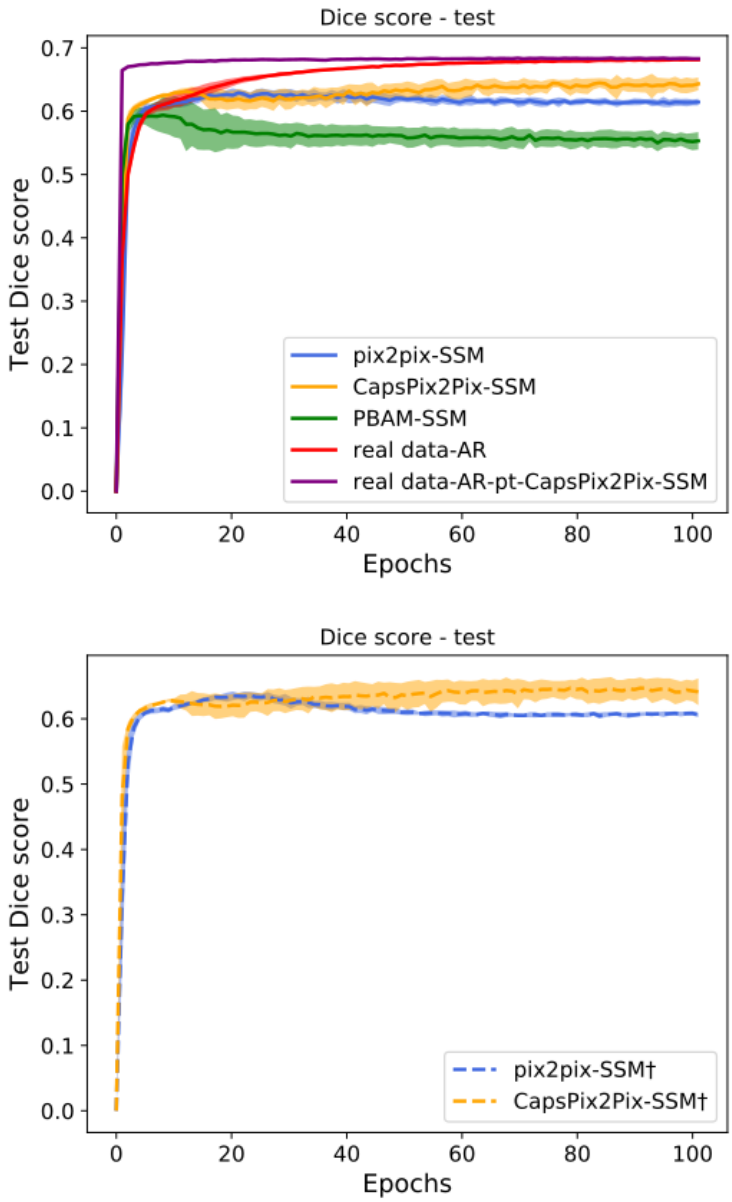

Figure A4: Comparison of U-net test curves for different datasets. The shaded regions represent ±1 standard deviations. Abbreviations: PBAM = physics-based imaging appearance model; SSM = (synthetic labels of the) statistical shape model; AR = augmented real (labels); pt = pretrained network.

Table A1: Per-experiment Dice scores for different datasets. Abbreviations: AR = augmented real data; P2P = `pix2pix`; Caps = CapsPix2Pix; pt = pretrained on CapsPix2Pix-AR data; Std = standard deviation.

Exp	AR	PBAM-SSM	P2P-AR	Caps-AR	P2P-SSM	Caps-SSM	AR-pt
1	0.6834	0.6121	0.6564	0.6358	0.6448	0.6558	0.6869
2	0.6829	0.5954	0.6578	0.6350	0.6332	0.6531	0.6867
3	0.6828	0.6154	0.6606	0.6398	0.6418	0.6537	0.6871
4	0.6839	0.6071	0.6602	0.6400	0.6423	0.6500	0.6864
5	0.6818	0.6021	0.6564	0.6389	0.6475	0.6529	0.6882
6	0.6821	0.5993	0.6583	0.6343	0.6394	0.6503	0.6876
7	0.6827	0.6089	0.6617	0.6270	0.6411	0.6490	0.6864
8	0.6808	0.6169	0.6568	0.6329	0.6394	0.6556	0.6867
9	0.6843	0.6129	0.6678	0.6342	0.6353	0.6578	0.6866
10	0.6820	0.6240	0.6560	0.6350	0.6426	0.6500	0.6875
Mean	0.6827	0.6094	0.6592	0.6353	0.6407	0.6528	0.6870
Std	0.0010	0.0083	0.0034	0.0036	0.0040	0.0028	0.0005

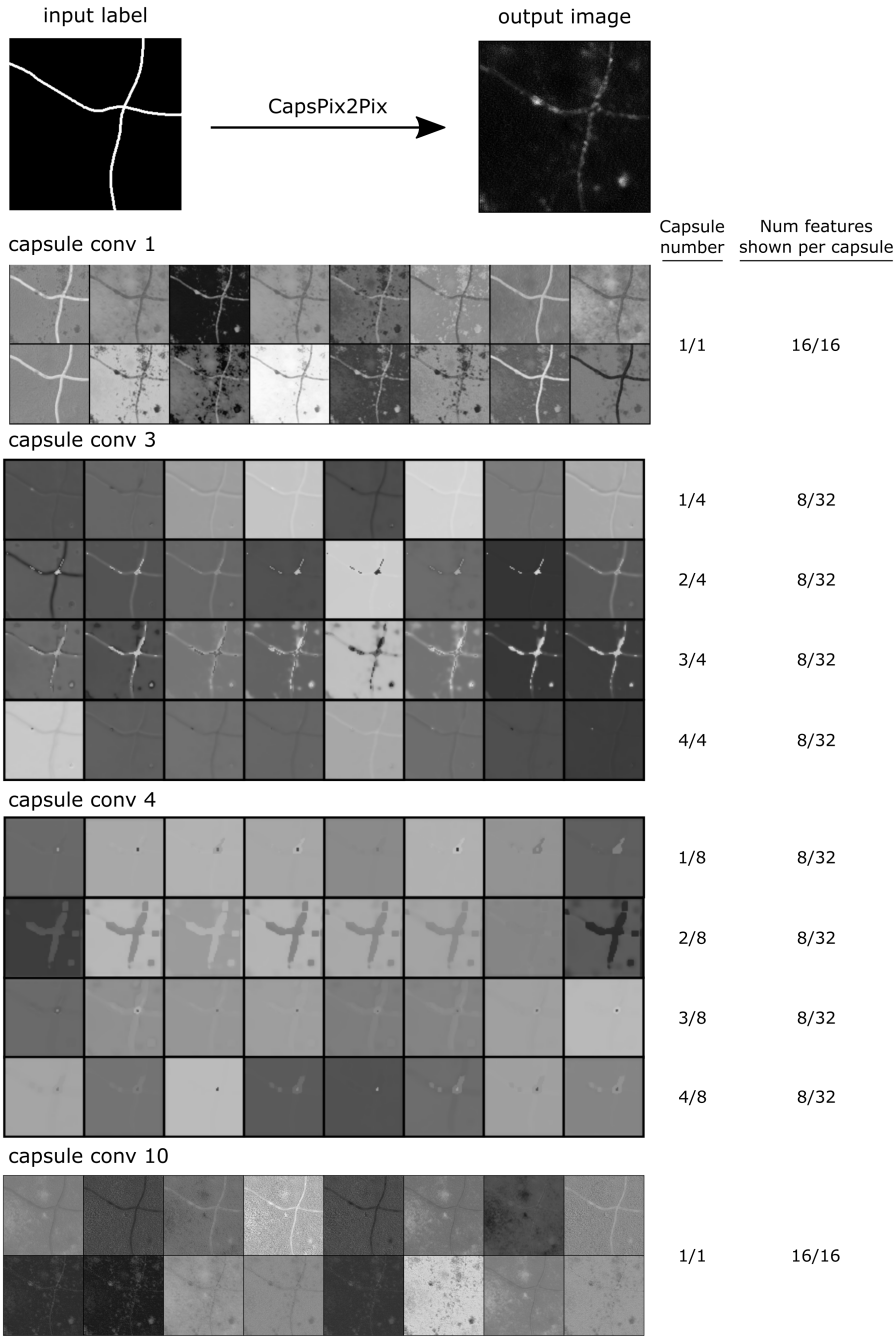

Figure A5: Examples of activations/ features at different capsule layers. The capsules from the intermediate layers group similar features (capsule conv 3 & 4). The outputs of the last convolutional capsules (capsule conv 10) are a combination of different types of features.

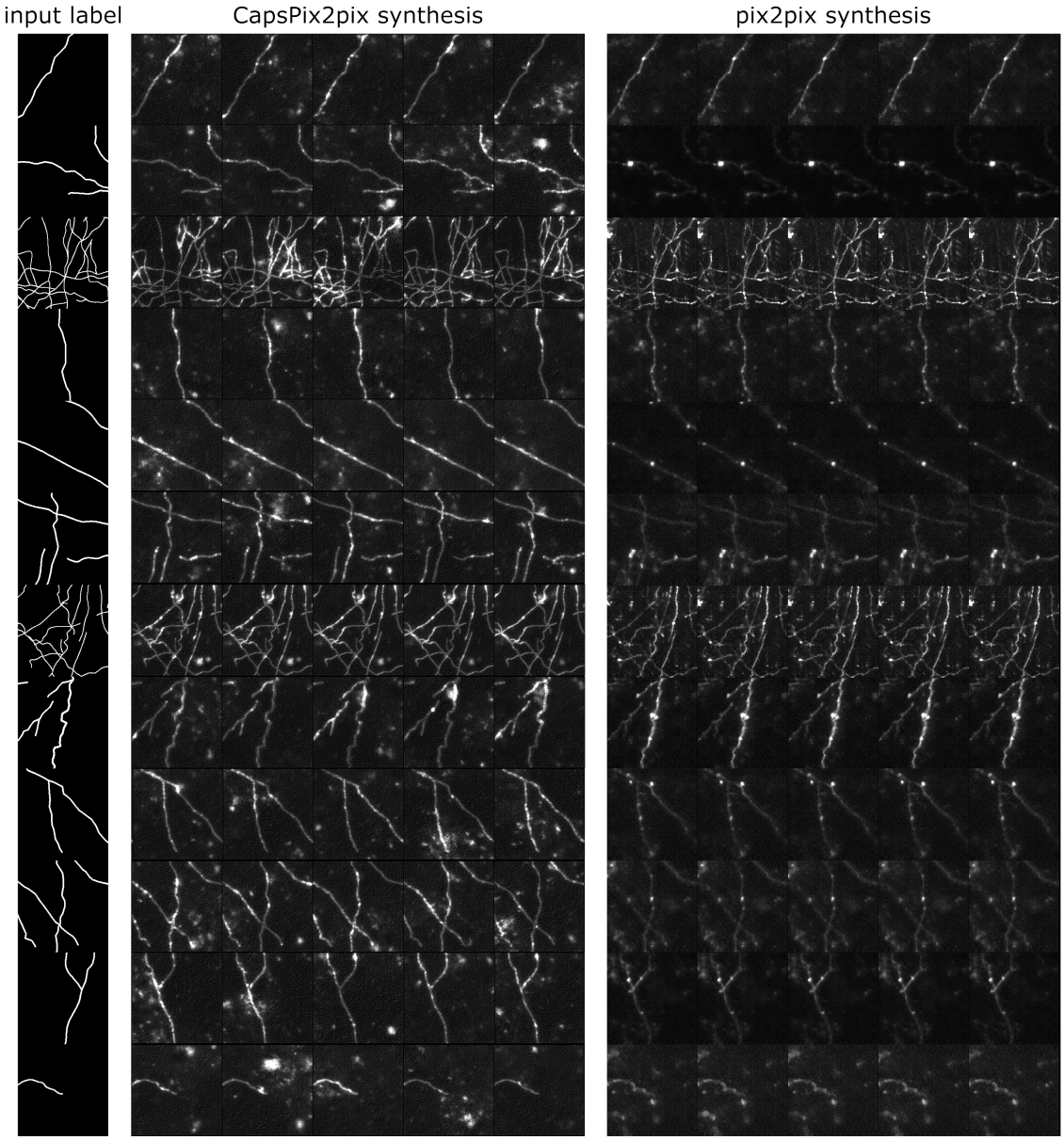

Figure A6: Examples of synthetic images from CapsPix2Pix and `pix2pix` based on the same labels. CapsPix2Pix has a large variation between synthetic images (when varying the z vector), but `pix2pix` only slightly alters the image (by randomly applying dropout during inference).

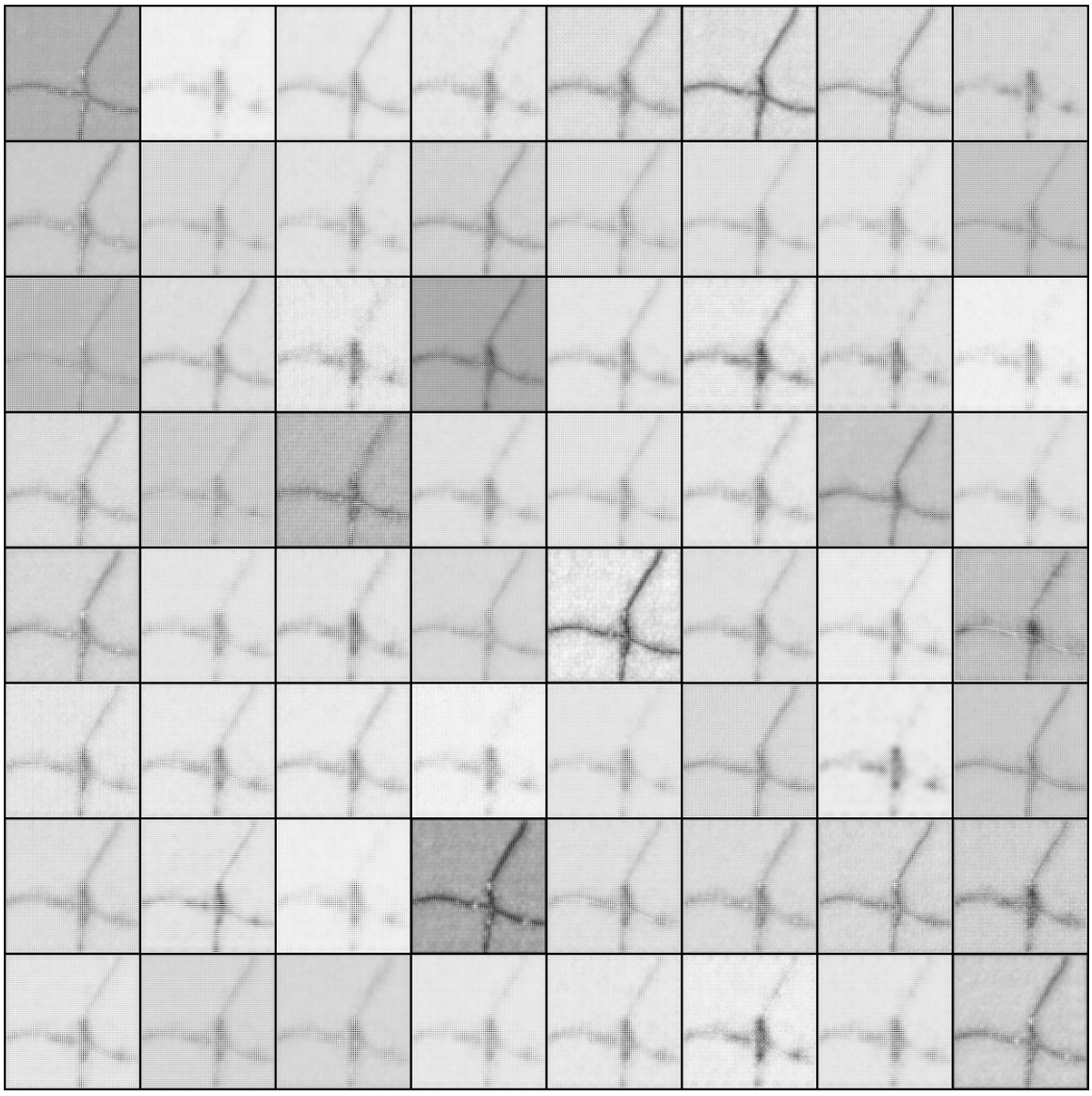

Figure A7: All 64 activations/ features of the last layer in a trained `pix2pix` model for a single input (the same as in Fig. 2).