## GAN with Spade

Uploaded new files to branch of lars

GAN\_spade\_esmee.py

VAE\_CIAN\_TRAIN\_esmee.ipynb

utils.py (I did not create this, but this version has the correct path in it compared to the one on the github of Cian)

Useful information about Spade regarding GAN

Paper: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8953676>

* Specific form of conditional image synthesis: converting a semantic segmentation mask to a photorealistic image.
* Spatially-adaptive normalization = a conditional normalization layer that modulates the activations using input semantic layouts through a spatially-adaptive, learned transformation and can effectively propagate the semantic information throughout the network.
* They apply it to a GAN
* Training dataset: paired segmentation masks and images 🡪 this is also our dataset
* Their normalization layer applies a spatially varying affine transformation, making it suitable for image synthesis from semantic masks.
* The activation is normalized in the channel-wise manner and then modulated with learned scale and bias
* Diagram

  Description automatically generated
  + Mask is first projected onto an embedding space
  + Then convolved to produce the modulation parameters gamma and beta of the normalization layer.
  + Gamma and beta are tensors with spatial dimensions (not vectors), and depend on the input segmentation mask and vary with respect to the location
  + The produced gamma and beta are multiplied and added to the normalized activation element-wise.
  + The functions gamma and beta are implemented using a simple two-layer convolutional network 🡪 details can be found in appendix of arXiv version
* Generator: there is no need to feed the segmentation map to the first layer of the generator, since the learned modulation parameters have encoded enough information about the label layout. Therefore, discard the encoder part of the generator 🡪 make sure we do this in our code
* Diagram

  Description automatically generated
  + This is the architecture of the generator.
  + Right: a series of spade residual (ResNet) blocks with upsampling layers
  + The modulation parameters of all the normalization layers are learned using spade
  + Left: structure of one residual block with spade
  + Since each residual block operates at a different scale, spade downsamples the semantic mask to match the spatial resolution
  + The generator is trained with the same multi-scale discriminator and loss function as used in pix2pixHD, but the least squared loss term is replaced with the hinge loss term 🡪 make sure this is also in our code
  + Only activations from the previous layer are normalized, hence it can better preserve the semantic information
* Implementation details 🡪 make sure this is used in our code
  + Learning rate generator: 0.0001
  + Learning rate discriminator: 0.0004
  + Apply Spectral Norm to all the layers in both generator and discriminator
  + Adam: beta1=0 and beta2=0.999

arXiv version of paper: <https://arxiv.org/pdf/1903.07291.pdf>

* The appendix contains more information on the architecture and implementation of the model.
* Generator:
  + Series of spade ResBlks with nearest neighbor upsampling
  + Spectral Norm applied to all the convolutional layers in the generator
  + Diagram

    Description automatically generated
  + In case the number of channels before and after the residual block is different, the skip connection is also learned
  + Diagram, table

    Description automatically generated
  + The semantic segmentation mask is passed to the generator through the proposed spade ResBlks.
* Discriminator:
  + Apply Spectral Norm to all the convolutional layers of the discriminator
  + Input is the concatenation of the segmentation map and the image
  + Table

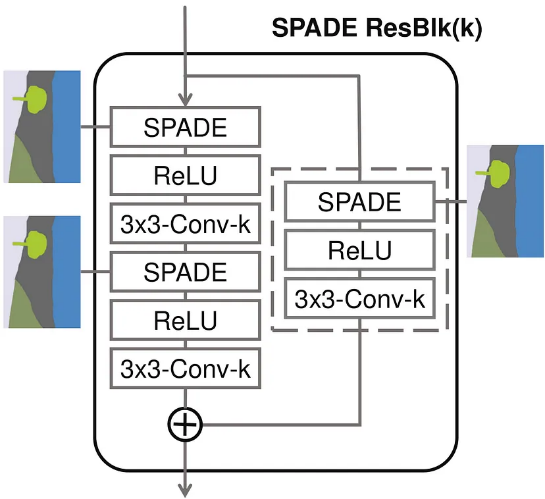
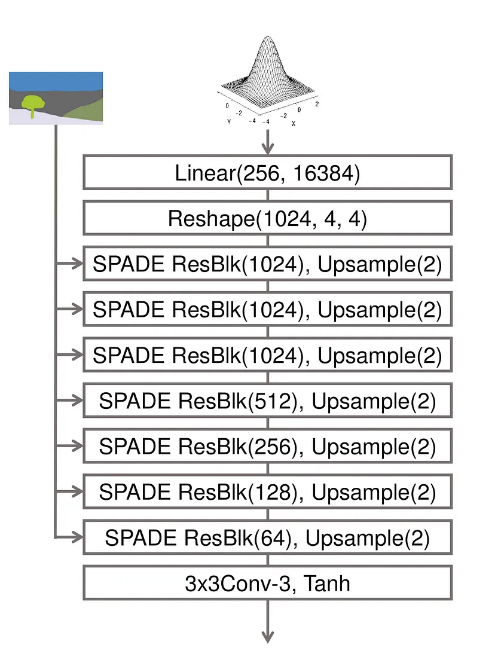
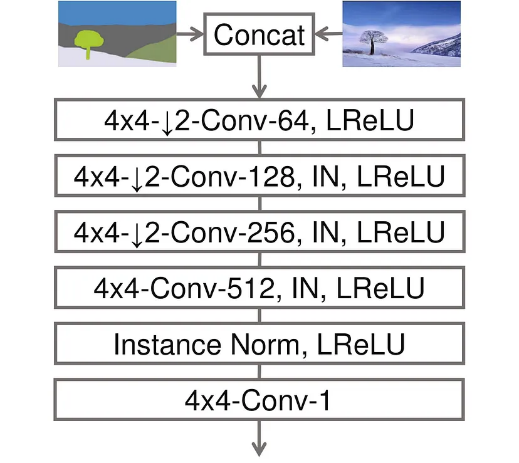
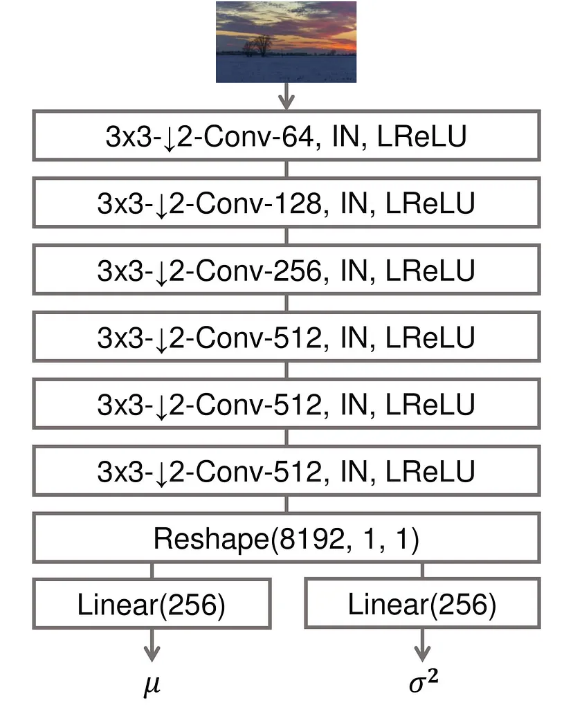
    Description automatically generated with medium confidence
* Image encoder: consists of 6 stride-2 convolutional layers followed by two linear layers to produce the mean and variance of the output distribution
  + Table

    Description automatically generated
* Learning objective: use the learning objective function from pix2pixHD work, but replace the LS-GAN loss term with the Hinge loss term. Same weighting among the loss terms in the objective function as that in the pix2pixHD work.
* Datasets: the image sizes are 256 x 256, except for cityscapes 512 x 256 🡪 our images are 64 x 64

Code of paper: <https://github.com/NVlabs/SPADE/tree/master/models/networks>

Website: <https://kushaj.medium.com/spade-state-of-the-art-in-image-to-image-translation-by-nvidia-bb49f2db2ce3>

* If you want your output image to resemble some other image (take the style of some image and add it to your output image), you need an image encoder which would provide the mean and variance values for the random Gaussian distribution 🡪 do we have that in our code?
* Every pixel value in your seg map corresponds to a class and you cannot introduce new pixel values.
* Diagram, schematic

  Description automatically generated
  + SPADE first resizes your seg map to match the size of the features and then we apply a conv layer to the resized seg map to extract the features.
  + To normalize our feature map, we first normalize our feature map using BatchNorm and then denormalize using the values we get from the seg map.
  + Code in website!
* 
  + Extension of the ResNet block
  + Skip-connection is important because it allows for training of deeper networks and we do not have to suffer from problems of vanishing gradients
  + Code in website!
* Generator: 
  + Code in website!
* Discriminator: 
  + Code in website!
* Loss function
  + 
  + Instead of computing the loss function on a fixed size of the image, you can compute the losses at different sizes of the image and sum them all. This would stabilize the training as the generator has to produce natural statistics at multiple scales.
  + So, extract features from multiple layers of the discriminator and learn to match these intermediate representations from the real and the synthesized images. This is done by taking features out of a pretrained VGG model. This is called perceptual loss.
  + See code in website
  + So we take the two images, real and synthesized and pass it through VGG network. We compare the intermediate feature maps to compute the loss.
* Weight initialization
  + This website: He. Initialization. See code in website.
  + Original paper: Glorot initialization.
* Image encoder
  + Final part of the model
  + Used if you want to transfer style from one image to the output of SPADE.
  + It works by outputting the mean and variance values from which we compute the random gaussian noise that we input to the generator.
  + Code in website!
  + 

Code of website: <https://github.com/KushajveerSingh/SPADE-PyTorch>

* Table

  Description automatically generated
* Since our images also have size 64x64, I am going to implement his code instead of the code from the original paper.
* He says that if you want to make your own model, you should follow the ‘train\_model.ipynb’ notebook
* A picture containing table

  Description automatically generated

Try to implement Spade in the GAN

Use this code: <https://github.com/KushajveerSingh/SPADE-PyTorch>

Dataloader -> cityscapes.py

* Compared this to our utils.py
  + Also consists of the def \_\_init\_\_() and def \_\_len\_\_(self) and def \_\_getitem\_\_(self, index)
  + The dataloader from the website contains more functions, but since the utils.py script is made by Cian, I am not going to change anything in there.

Train\_model.ipynb

* <https://github.com/KushajveerSingh/SPADE-PyTorch/blob/master/src/notebooks/train_model.ipynb>
* This contains code we also have in the training jupyter notebook
* And it also contains the models, which we create in the separate GAN .py file
* Loading the data goes correct in our model I think
  + The image and segmentation size of our model are [31, 1, 64, 64] while the images and segmentations from the dataset used by the website have size [4,3,64,64] and [4,1,64,64].
  + The website shows the image and segmentations separately as images, but we overlay them in our code.
* Class Args:
  + This is not in our current model, so I put this into the GAN\_spade\_esmee.py file
  + It is used further on in the model, so we need it
  + Website:
  + Graphical user interface, text, application

    Description automatically generated
  + Implementation in our code:
  + Text

    Description automatically generated
  + I have used the exact same numbers because the image sizes of the cityscape dataset are the same as our dataset
* Class Generator(nn.Module):
  + Uses the SPADEResBlk 🡪 check the code & create spade\_resblk.py file in which SPADEResBlk is defined & import them into the GAN py file
  + For the first time trying this generator, I will use the same numbers as the website because the image sizes of our dataset are the same
  + Website:
  + Text

    Description automatically generated
  + Implementation in our code:
  + Text

    Description automatically generated
  + The class Generator in the jupyter notebook of the website is different than the class SPADEGenerator in the generator.py file in models folder
  + Text

    Description automatically generated
* Class Discriminator(nn.Module):
  + For the first time trying this discriminator, I will use the same numbers as the website because the image sizes of our dataset are the same
  + Website:
  + Text

    Description automatically generated
  + Implementation in our code:
  + Text

    Description automatically generated
  + The custom models are only defined in the discriminator.py file of the website:
  + Graphical user interface, text, application, email

    Description automatically generated
  + These custom models differ from the ones we used before, so I implemented those in the py file as well
  + The class Discriminator in the jupyter notebook of the website is different than the class SPADEDiscriminator in the discriminator.py file in models folder
  + Graphical user interface, text, application

    Description automatically generated
* Def weights\_init(m):
  + Website:
  + A screenshot of a computer

    Description automatically generated with medium confidence
  + This is the same code as what we used before
* Next cell:
  + Website:
  + Text

    Description automatically generated
  + The code we used before has the same structure and there are minor differences. This is the code I implemented:
  + Graphical user interface, text

    Description automatically generated
  + Change the number of epochs when trying if the code works!!!
  + With the GANLoss as defined in ganloss.py of the website. I have put this in our jupyter notebook:
  + Text

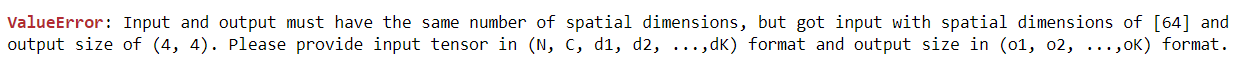
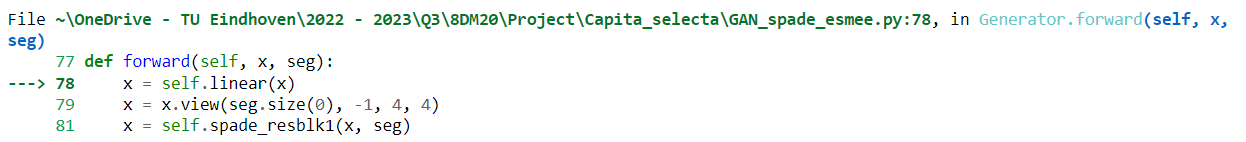
    Description automatically generated
* Training:
  + Website:
  + Text

    Description automatically generated
  + I have implemented exactly this code in our jupyter notebook. It is almost identical to the code we used before. 🡪 see below the newest implementation
  + In this implementation I left out the saveModels function, but when it works this should be implemented again.
* Copy and paste the last cells of the jupyter notebook of the website:
  + Graphical user interface, text, application, email

    Description automatically generated
  + Text, application

    Description automatically generated
  + See if we want to implement this in a different way.

Running this implementation of the code

* First error in the fake\_img=gen(noise, seg) line during training
  + Referred to both Generator.forward and SPADEResBlk.forward
  + 
  + First solution to try: adjust the size of the noise. The notebook from the website used torch.rand(4, 256) but in our own code we used torch.randn(32, 255) so I used that value again.
    - Again an error in fake\_img = gen(noise, seg)
    - 
    - 

Training

* Instead of the implementation described before, Paula and I think we should use img instead of seg for both the discriminator and the generator!
* Text

  Description automatically generated
* The rest of the implementation is the same as the code from the website.
* I did not yet check if the Generator and Discriminator in the GAN\_spade\_esmee.py file need to be adjusted to match this implementation!
* I discussed with Paula why we think we should not use the code of the website for this part: using seg would result in masks that are created instead of images. Therefore use img to also generate the prostate MR images.
  + The website also says they want to generate images with the GAN, but their segmentation maps contain much more classes, so the segmentation mask contains more information that the GAN can use to create an image out of it.