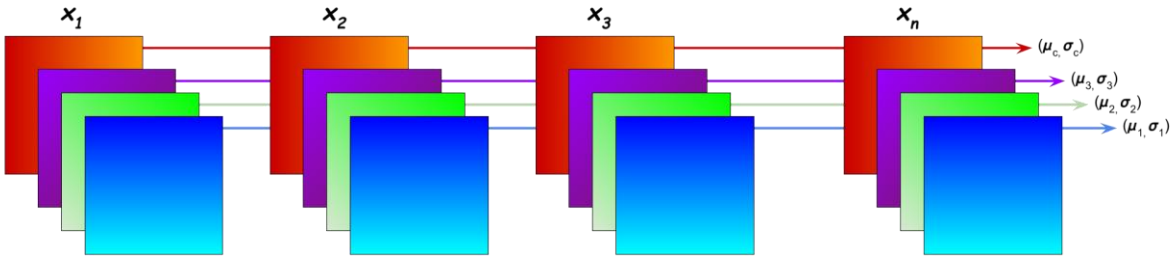
Project Idea 1 – Encoding information into normalization layers:

- **Some background on normalization layers:**

Normalization layers are simple operations that apply the following transformation on the feature maps: $\hat{x} = \gamma_s \frac{x-\mu_c}{\sqrt{\sigma_c^2+\epsilon}} + \beta_s$ where X: input feature map; μ_c: a calculate mean; σ_c^2: variance; γ_s, β_s are learned parameters. The calculation of the mean and standard deviation depends on the type of normalization. For the case of batch normalization, for example, the calculation for the mean and standard deviation is shown pictorially below: (I don't want to put more formulas in here :P)



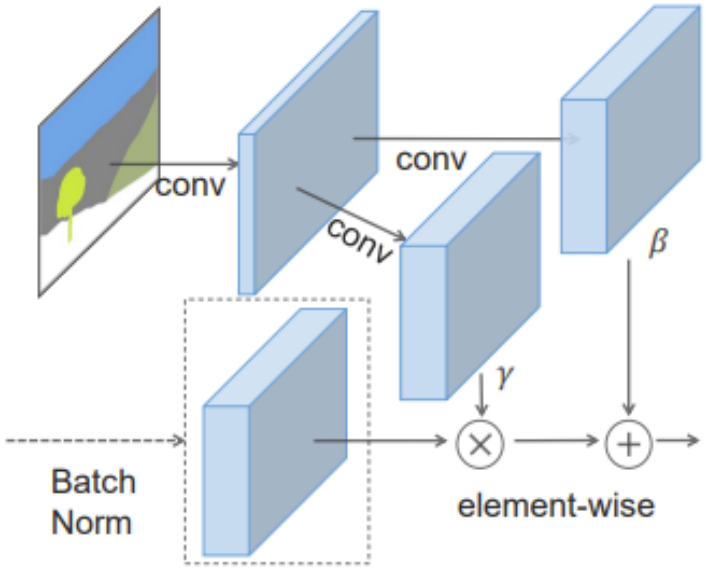
If you want more information about normalization layers, you can check this link. It is very well explained here.
[Batch Normalization, Instance Normalization, Layer Normalization: Structural Nuances | Becoming Human: Artificial Intelligence Magazine]

- **Why normalization layers?**
  As far as I know, the theory behind why normalization layers work is still poor (there is going to be a lecture this week, we can perhaps learn more). The theories right now suggest that they may help in: (optional – can skip this for now if you want)
  - Better behaving gradients – the gradient distribution has a lower variance
  - Converts a low-rank model to a full rank model

  Empirically, using normalization layers has always shown improvement in the performance of neural networks.

- **What can we do with these layers?**
  I mentioned in the first bullet point that we have two optional learnable parameters in normalization layers: namely γ_s, β_s. A problem with normalization layers is that they usually remove semantic information (Park et al). However, we can also encode information in these two.

How? For example, in a paper by NVIDIA, information about segmentation masks was encoded in these γ_s, β_s. Look at the following picture:



In a few words:

1. First normal Batch norm is used: $\frac{x-\mu_c}{\sqrt{\sigma_c^2+\epsilon}}$

2. Then we apply convolutional layers to segmentation masks and extract γ_s, β_s. Then we multiply with γ_s, and add β_s to get the final normalized layer according to the formula I gave above: $\hat{x} = \gamma_s \frac{x-\mu_c}{\sqrt{\sigma_c^2+\epsilon}} + \beta_s$

Here is a link to the paper: 1903.07291.pdf (arxiv.org)

- **So what could we do?**

Now, we have a tool to encode additional information in a neural network through the Batch norm.

In my semester project, for example, I also used segmentation masks, but in addition, I used binary vectors to generate different medical image contrasts. With the binary vectors, I restricted the degrees of freedom of the newly generated images. Here is a schematic of the new normalization layer used:

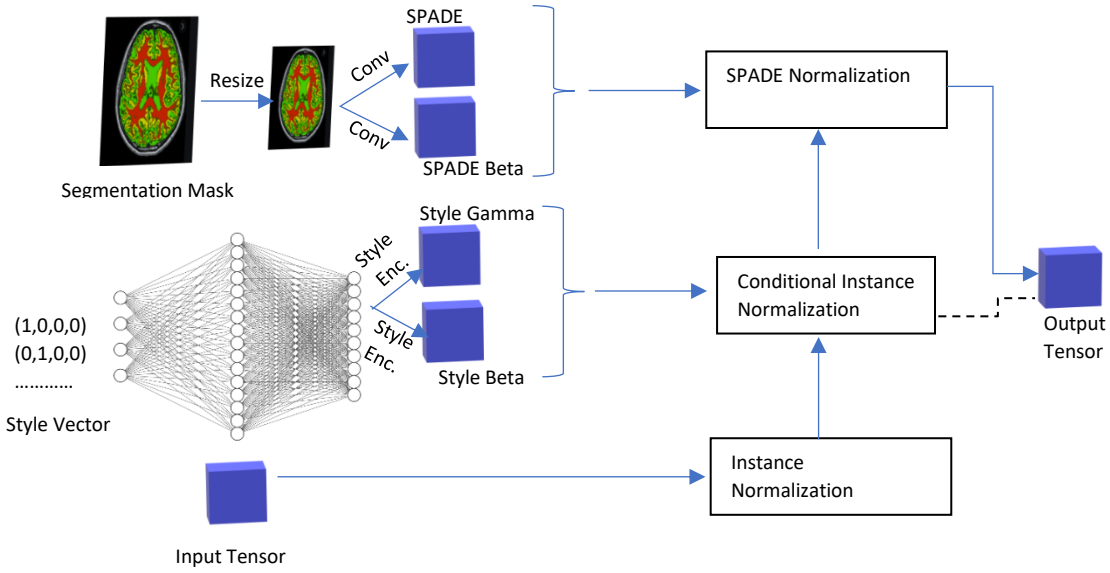


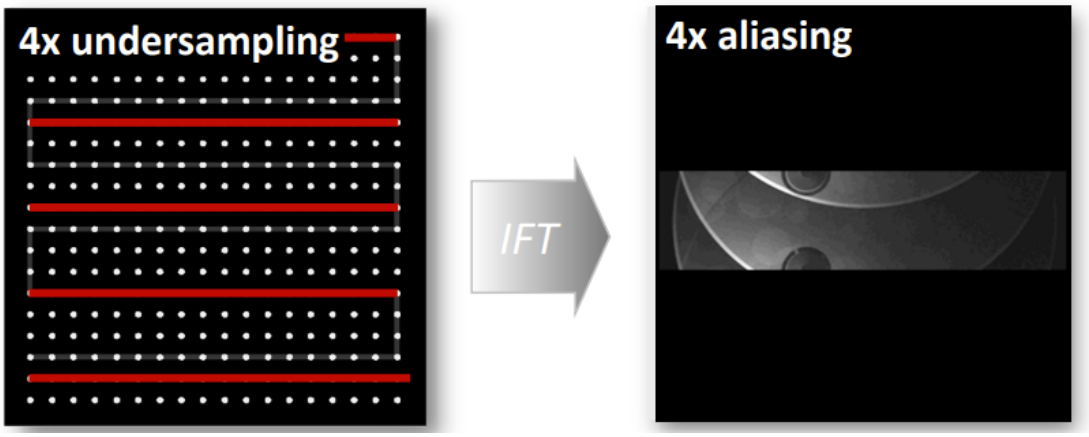
Figure 5: The style adaptation normalization layer.

So, we can use normalization layers to encode prior information. **For example, Jakub's idea about color palettes could be interesting.** We could encode color palettes into normalization layers which will serve as prior information to guide our GANs or VAEs.

Or, we could use another similar idea as well where we can use priors. Feel free to add new suggestions.

Project Idea 2 – Using model-based neural networks for image reconstruction

- **Simple background in MRI**
  In MRI, we acquire the signal in the Fourier transformed 2D space (known as k-space). The signal is usually acquired sequentially line-by-line in k-space. Then if we disregard the noise, we can obtain our image by just inverse Fourier transform.
  However, it usually takes a tremendous amount of time to cover all of this k-space, so we don't acquire all of it. Below, is shown the case when we undersample by a factor of 4. So out of 4 lines in k-space, we just acquire one.



As seen above, if we naively inverse Fourier transform this, then we get aliasing. This is explained by signal processing theory (Remember Shannon's sampling theorem).

- **So how do we get both: quality of images and less time?**
  We can formulate the above problem as an optimization problem. In this optimization problem, however, we need to add some prior information about the image. This is necessary because we need to account for the missing information of the k-space. This idea is known as compressed sensing. For example, we can assume that the derivatives are sparse in the image, so our optimization problem will be formulated as shown below:

$$\hat{\mathbf{x}} = argmin\_\mathrm{x}|\mathbf{MFx} - \mathbf{s}|_2^2 + \lambda|\nabla\mathbf{x}|_1$$

  - The first term: $\mathbf{MFx} - \mathbf{s}$**:** data consistency term-
    s: our under-sampled k-space that we have.
    M: will show which lines we got from k-space
    F: Fourier transform matrix
    X: our image, what we will try to reconstruct
  - The second term: $\lambda|\nabla\mathbf{x}|_1$ – imposes sparsity prior on the images we will try to reconstruct

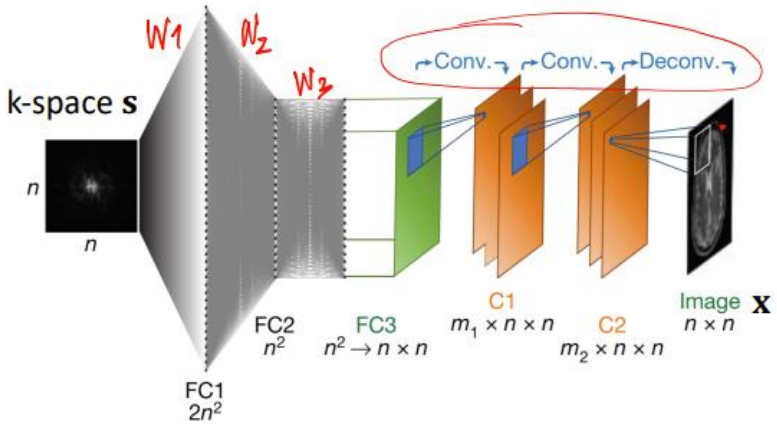
  Solving the optimization problem above has shown great improvements in the quality of the reconstructed images.

- **Why use such information in neural networks?**

  As Jakub asked, why do we not just feed this k-space into the network and ask it to do this reconstruction for us? This has been done as shown below. (the AUTOMAP architecture)

AUTOMAP

- Network $\mathcal{V}_{\boldsymbol{\theta}}$: 3 fully connected layers and 3 convolutional layers



  Idea: uses three fully connected layers in the beginning because to invert the Fourier representation we need all of k-space and not just local information provided by convolutional layers. Then we can apply some convolutional layers.

  Drawback: a lot of parameters especially due to the fully connected layers in the beginning. The images are huge and medical datasets are limited. (There could be other reasons as well)

We can decrease the number of parameters of such a network if we can decrease the degrees of freedom. That's why we can use neural networks that are informed about the model of the images that we are trying to reconstruct and reduce the number of parameters considerably.

- **How to construct such a network which will know the optimization problem shown above?**

  Before that, how do we solve the optimization problem?

$$\hat{\mathbf{x}} = argmin\_\mathrm{x}|\mathrm{A}\mathbf{x} - \mathbf{s}|_2^2 + \lambda|\mathrm{D}\mathbf{x}|_1$$

  We can use gradient descent to solve this. And the iteration will look as shown below:

$$\mathbf{x}^{(\mathbf{k+1})} \leftarrow \mathbf{x}^{(\mathbf{k})} - \alpha^{(k)}\, \mathbf{A}^{\mathbf{H}}\left(\mathbf{A}\mathbf{x}^{(\mathbf{k})} - \mathbf{s}\right) + \lambda \mathbf{D}^{\mathbf{T}} sign\left(\mathbf{D}\mathbf{x}^{(\mathbf{k})}\right)$$

  We have one problem if we want to use deep learning for this:

  - Sign → not differentiable so can't use this in a neural network. Instead, we relax this with $\sigma(t) = c_1\left(\frac{1}{1+\exp(-c_2 t)} - 0.5\right)$
    This function is close to the sign but it is differentiable.

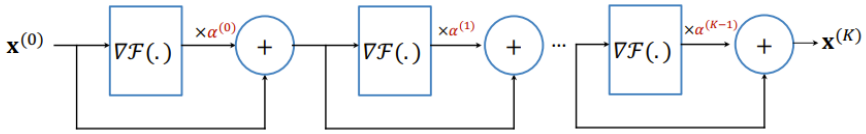
We add some further degrees of freedom to this:

- Instead of just the gradient filter D, we can add multiple filters that can be learned by the network(this will be our convolutional layers in the neural network. This is an advantage to the fixed classical way we use.

Now, let's concatenate all the gradient iterations. Our neural network will look as shown below:

$$\mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)} - \alpha^{(k)}\left[\mathbf{A}^{\mathrm{H}}\left(\mathbf{A}\mathbf{x}^{(k)} - \mathbf{s}\right) + \mathbf{D}^{T}\sigma\left(\mathbf{D}\mathbf{x}^{(k)}\right)\right]$$

- Can be seen as a neural net



Note: This network is called a variational network and it has considerably fewer parameters (0.15 M) compared to AUTOMAP (2000 M). It also gives very good reconstruction results.

- **What can we do utilizing this idea?**
  For one, we can include different priors which may lead to different gradient descent iterations and hence different neural network architectures.
  1) A proposed project is for example reconstruction of 3D heart imaging. We know that the pixels in most of the frames will not move. The prior we can use here is the vectorial total variation penalty as:

$$\sum_{p \in \text{ spatial locations}} \sqrt{|\nabla \mathbf{x_1}|_p^2 + \cdots + |\nabla \mathbf{x_T}|_p^2}$$

The formula above says: let's take spatial derivatives of the images in all frames, group them and say that they are sparse.
As I showed above, we will take the gradient descent and we need the gradient of this. We can then implement something similar to the architecture above. We can also play with other priors depending on our problem of interest.

2) Another thing: Christian mentioned at one time that he had some experience in the field of Ultrasound. We can construct a suitable model for Ultrasound as given in this paper: IEEE Xplore Full-Text PDF:
Depending on the speed that this sound propagates we do some reconstruction. We can construct a model which will explain how ultrasound is propagated and measured in the same form:

$$\min_{\mathbf{x}} \|\mathbf{L}\mathbf{x} - \mathbf{s}\|$$

I think the particular paper did not use any regularization term, it just used gradient descent iterations on the data term given above. Ultrasound is especially ill-posed because the transducer is usually from one side and some depth information is lost. So suitable priors can be used to deal with this problem efficiently. We can discuss more if you would like to follow this path. I can also get some help from a post-doc working on something similar if you would like.

3) If you know some problem, for which the classical problem solved by gradient descent works quite well and the deep learning approach is quite expensive (i.e. has a lot of parameters) you can drop some suggestions. We can blend the two. 😊