Neural Network design

In addition to learning *how* to use Neural Networks, we hope this course has stimulated your curiousity.

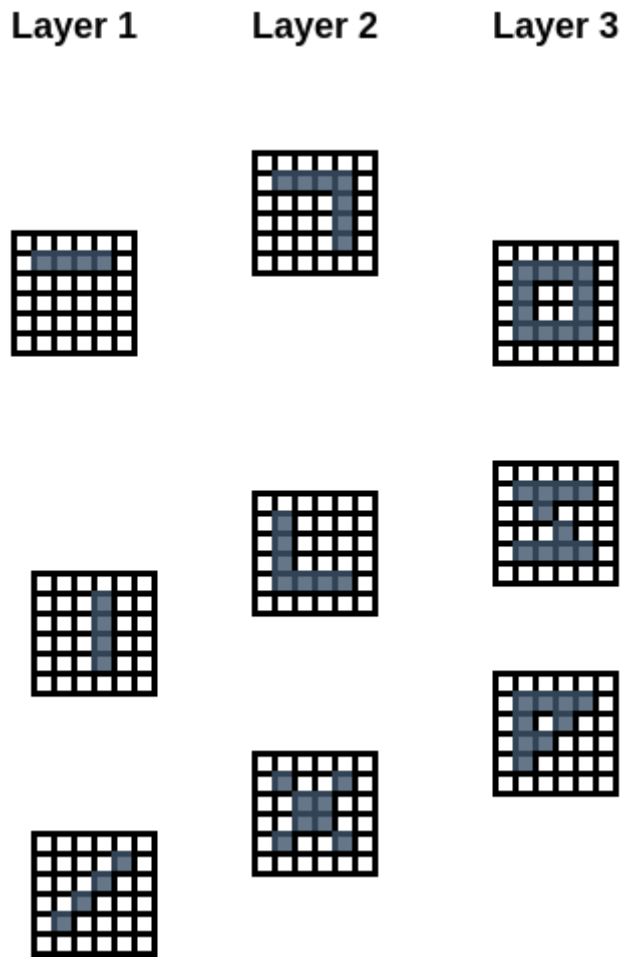
What strikes me as particularly curious:

- Neural Networks create representations allowing the Head Layer to solve a Classical ML task
- We don't know *how* these representations are created
- We certainly have not given any explicit instruction or requirement
- Yet the representation seems to be both useful for a particular task
- And *Transferable* to other tasks

We saw how a Neural Network for a vision task

- Seems to learn complex concepts
    - "Dimensions of meaning"
    - From smaller parts

Features by layer



So too we saw that Word Embeddings seem to learn dimensions of meaning.

Both Images and Text feel like complex domains

- Yet a mechanical process (training on examples) seem to "discover" meaning
- Without explicit direction or explanation of the domains

Another curiousity: complex tasks (e.g. NLP, image recognition) are solved with *simple* programs.

The "art" of Neural Networks is *not* highly skilled programming but instead

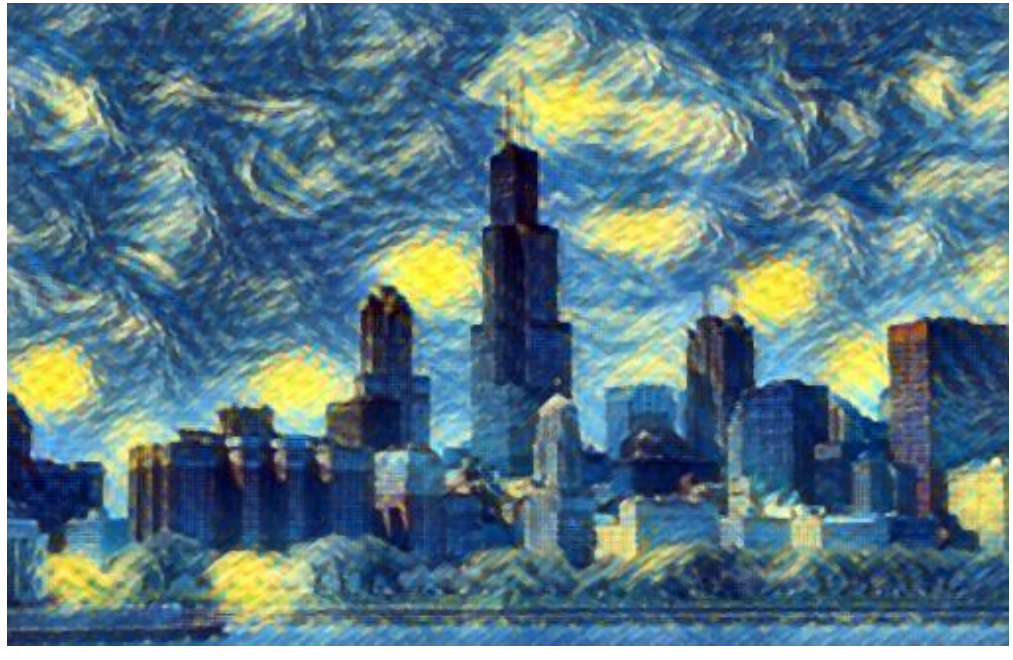
- Being clever and diligent in acquiring enough training examples
- Creating a Loss Function that captures the essence of the problem

For example: Neural Style Transfer is a task that

- Takes one image (the "Content Image")
- And an artistic style, as expressed by a "Style Image"
- Produces a new image that re-expresses the Content Image in the style of the Style Image

Content Image	Style Image
	

Generated Image



The "trick" in solving this task is in writing the Loss function

- Not in designing the network
- Once the Loss function has been created
- We then apply the skills we learned to minimize Loss Functions
- And the task is solved

Without going into detail the Loss Function has two parts

- A "content loss": the generated image $\vec{\mathbf{x}}$ should be close to the Source Image $\vec{\mathbf{p}}$
- A "style loss": the style of the generated image $\vec{\mathbf{x}}$ and the Style Image $\vec{\mathbf{a}}$ should be close

$$\mathcal{L} = \mathcal{L}_{\text{content}}(\vec{\mathbf{p}}, \vec{\mathbf{x}}) + \mathcal{L}_{\text{style}}(\vec{\mathbf{a}}, \vec{\mathbf{x}})$$

To be sure: there is some cleverness involved in

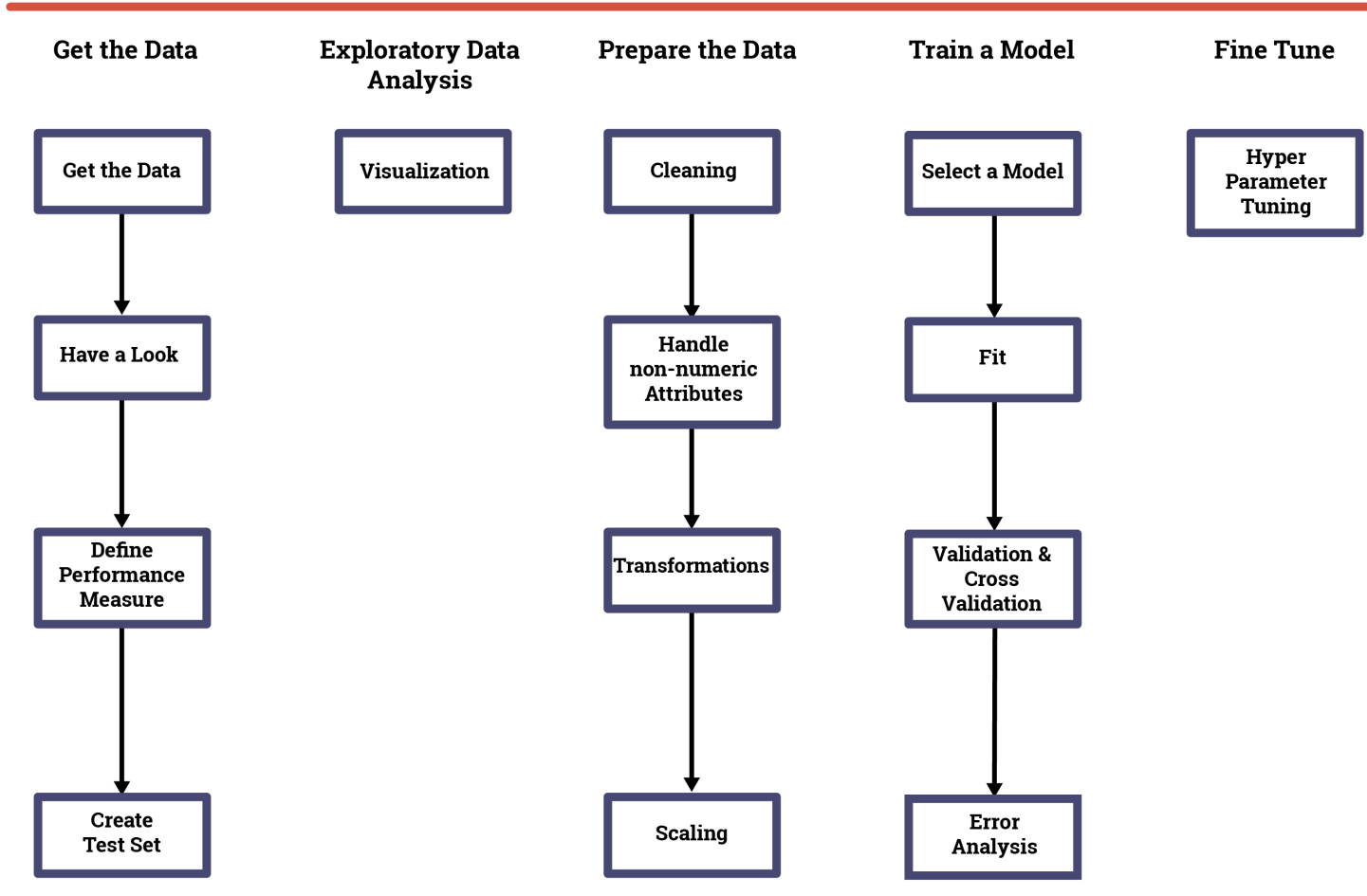
- Defining what "style" is
- What is the best measure of "being close"

but given the framework of the Loss Function, these tasks are closer to Engineering than Art.

Thus, the simple skills we learned

- When applied with discipline (the "Recipe for Machine Learning")
- Can solve seemingly complicated tasks
- Once we have defined a Loss Function embodying our objectives

Recipe for Machine Learning



When these skills are combined with Transfer Learning

- You are truly able to "stand on the shoulder of giants"
- And hopefully solve those tasks that are meaningful to your domain

We look forward to the day when *you* will be the giant on whose shoulders others stand.

In [2]:
```
print("Done")
```

```
Done
```