Deep learning applications to solar flare observations

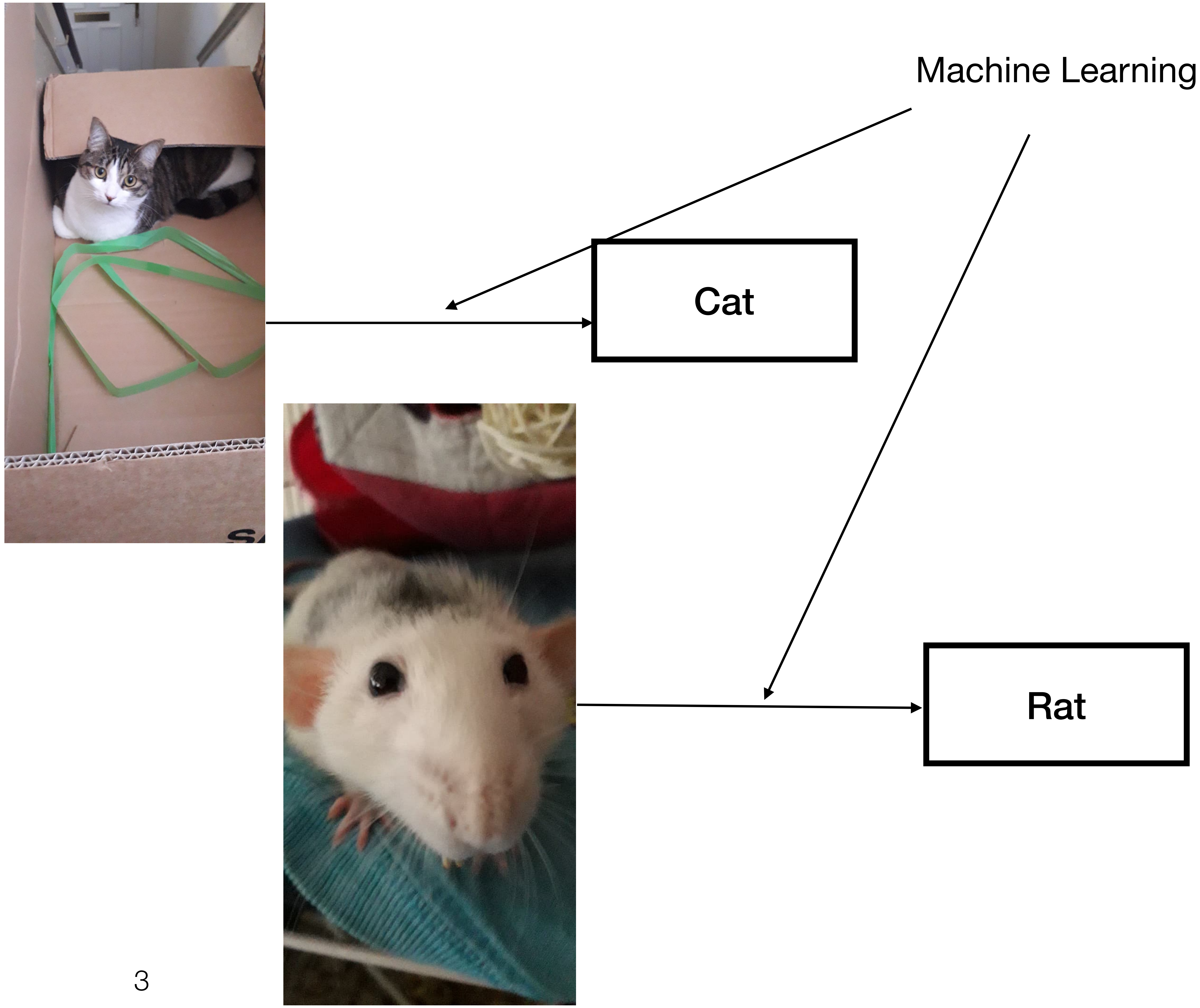
John Armstrong

Overview

1. Introduction to machine learning
    - different types of machine learning
2. Introduction to deep learning
    - perceptron
    - stacking perceptrons and layers
    - deep neural networks
3. Example of using a deep network for image classification
4. Correcting for seeing artifacts in flare observations with deep learning
    - GANs
5. Chromospheric solar flare inversions
    - INNs

What is machine learning?

The process of using statistical techniques to give computers the ability to learn how to perform a specific task *without* being explicitly programmed



Different Types of Machine Learning

	Supervised	Unsupervised
Classical	Decision Trees, SVMs, shallow neural networks	Clustering, Dimensionality Reduction
Deep	Multi-layer ANNs/ CNNs, ResNets	GANs, VAEs, INNs

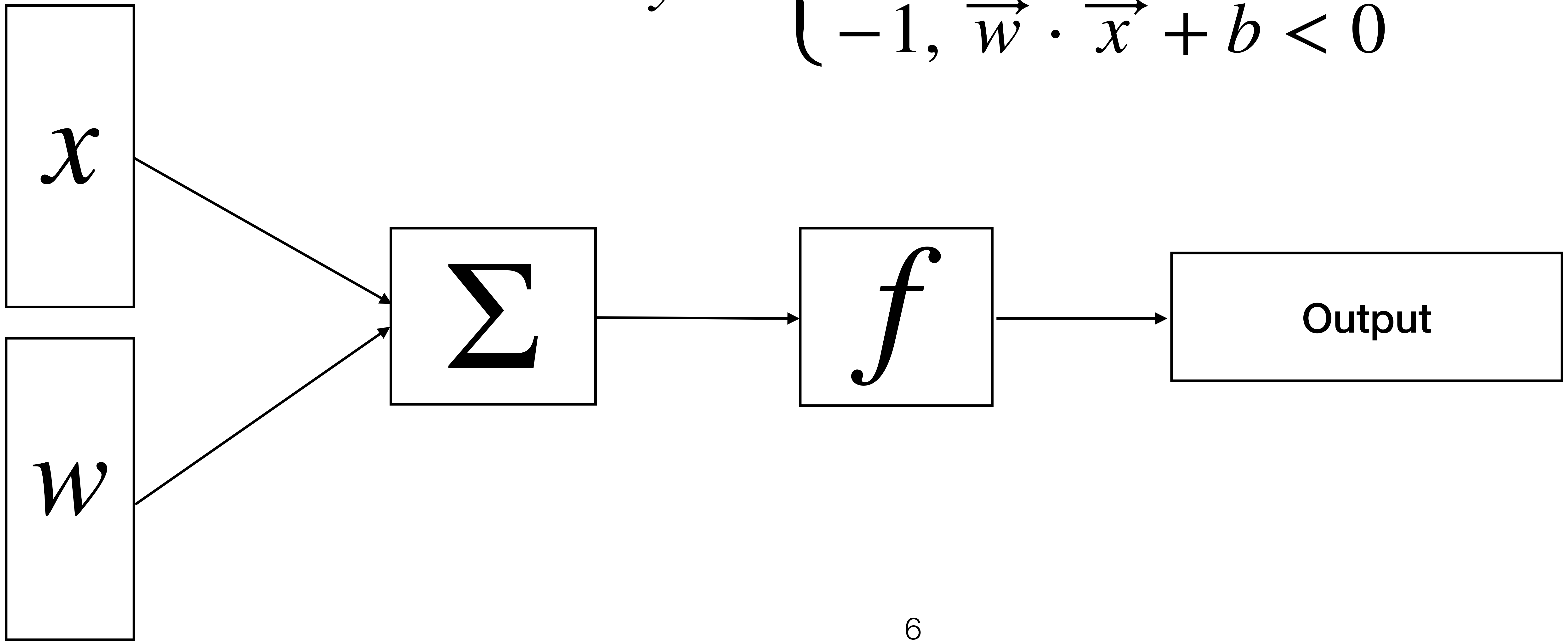
Different Types of Machine Learning

	Supervised	Unsupervised
Classical	Decision Trees, SVMs, shallow neural networks	Clustering, Dimensionality Reduction
Deep	Multi-layer ANNs/ CNNs, ResNets	GANs, VAEs, INNs

Rosenblatt's Perceptron

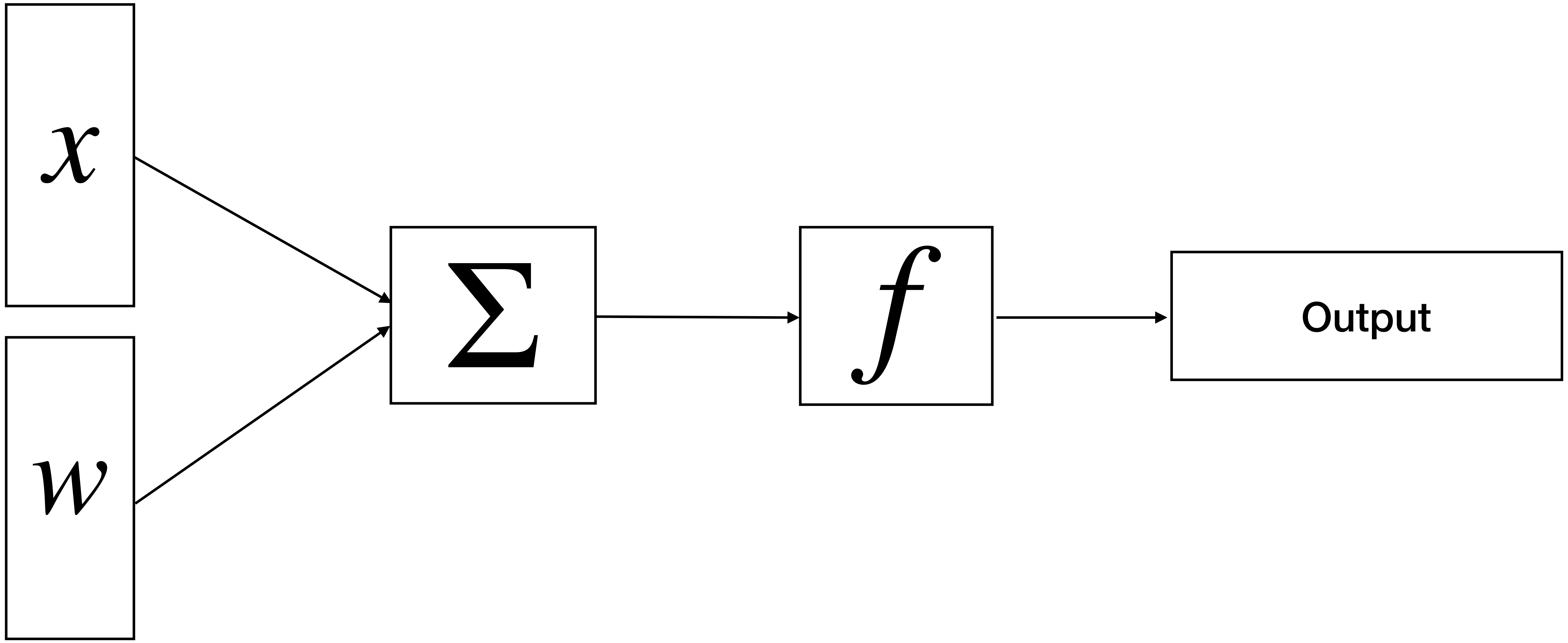
- Based on the model of the biological neuron (of the time).
- Binary classification/regression only. Unable to solve non-linear problems.
- Mathematically simple.

$$y = \begin{cases} 1, & \vec{w} \cdot \vec{x} + b \geq 0 \\ -1, & \vec{w} \cdot \vec{x} + b < 0 \end{cases}$$



Rosenblatt's Training

$$w_i(t) = w_i(t-1) + \eta(d_i - y_i)x_i$$



Modern Perceptrons

- The combination of input and weights can be achieved with any linear combining function

- The choice of the non-linear function has been extended to a smooth continuous function allowing for derivatives to be found

- A new training scheme known as backpropagation is introduced which is achieved through stochastic gradient descent (SGD)

Backpropagation

- This is a generalisation of Rosenblatt’s training technique.

$$w_i(t) = w_i(t-1) + \eta \nabla_w L(w;x)$$

New, updated weight for current timestep

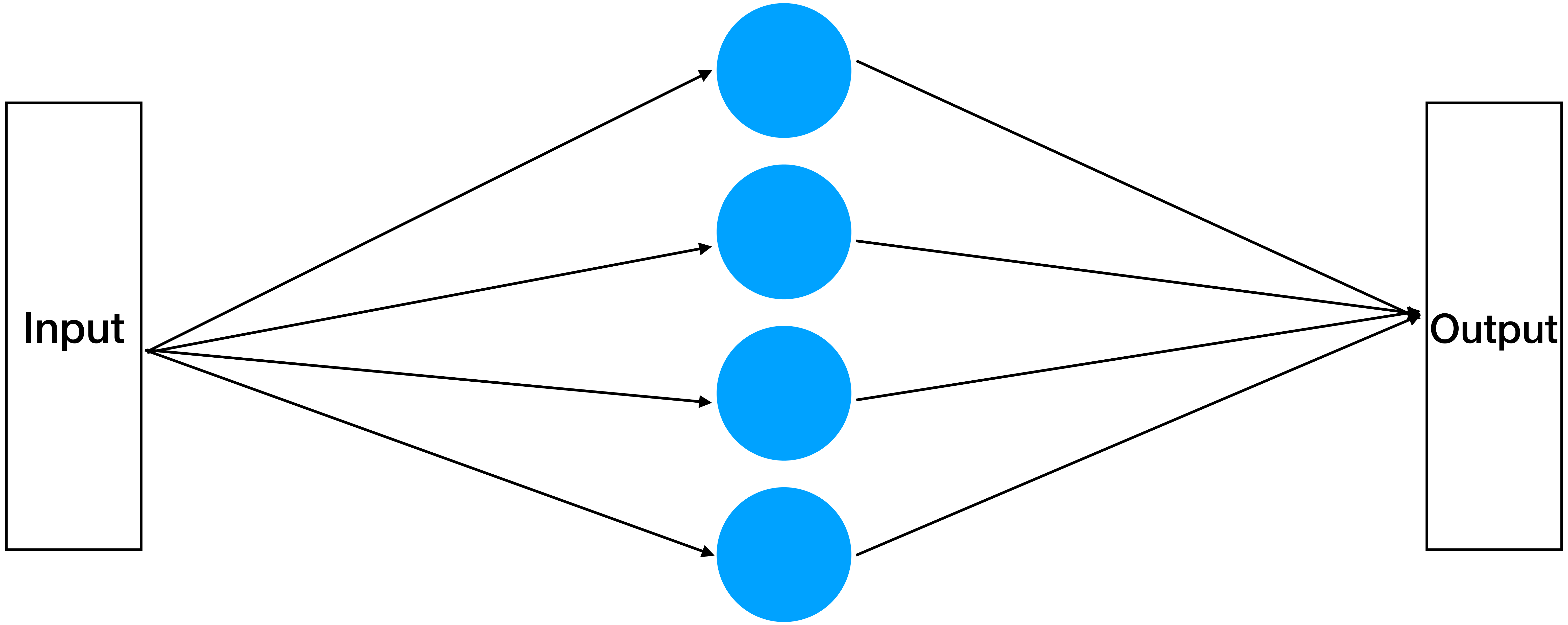
Weight from previous timestep

Learning Rate

Loss function

Stacking Neurons

- We can approximate complex non-linear functions by passing our data through a series of neurons simultaneously with different weight vectors and reconsolidating them at the other end



Layer of Neurons

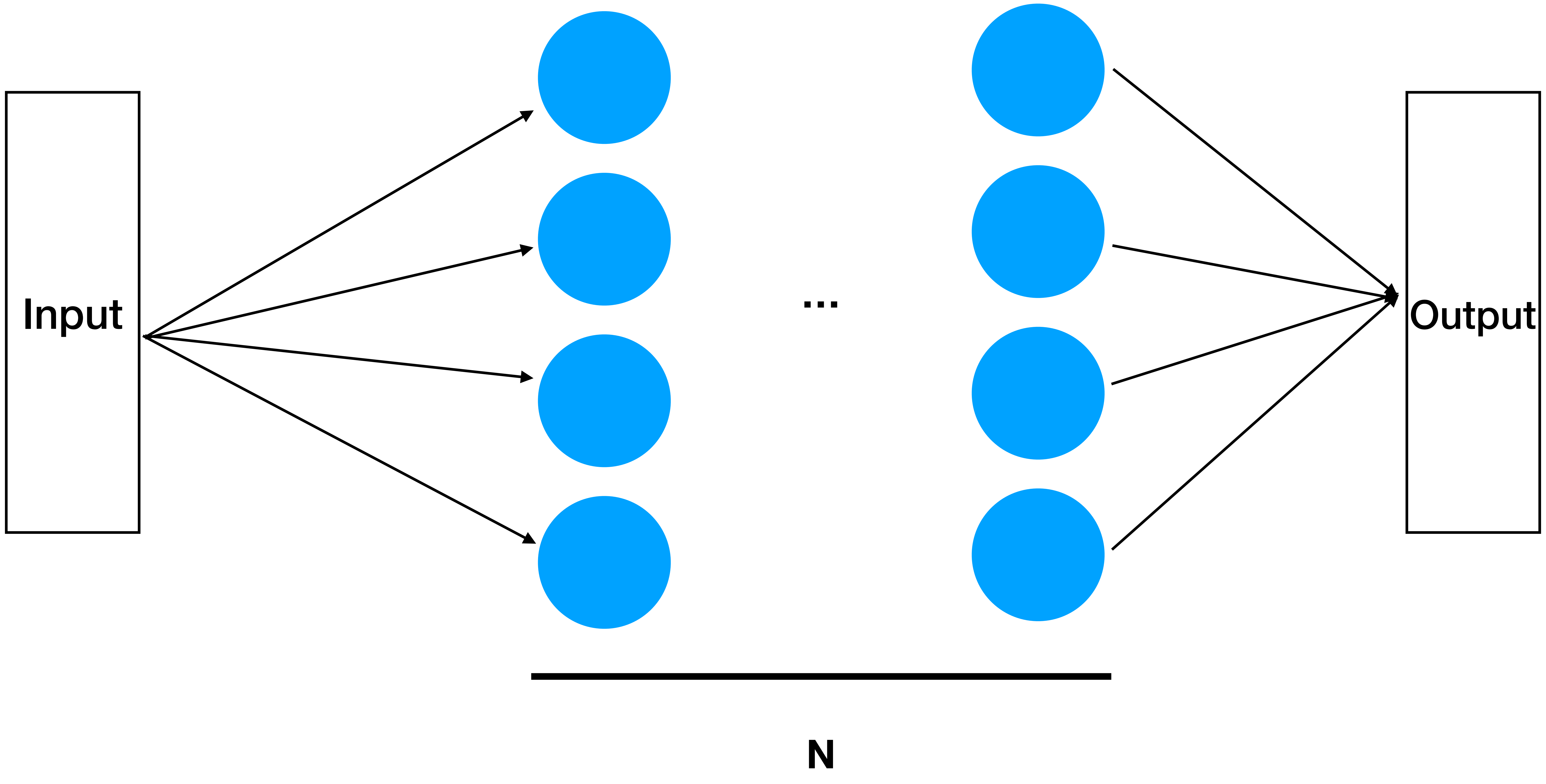
Limitations

- Can theoretically learn any problem given sufficient time
- Many problems are too complex to be learned efficiently with one layer
- Can make the layer wider (more neurons) to approximate more complex functions but this is comes at a cost of memory

Deep Learning

- Deep learning = machine learning done with deep neural networks
- Deep neural network = a neural network composed of >1 layer of neurons
- Deep learning is the workaround to efficiency in training for complex functions
- Increasing the number of layers increases the rate of convergence

Deep Neural Networks



Types of Deep Networks

1. Fully-connected

- Every neuron in the preceding layer is connected to every neuron in the current layer
- Linearity in the neuron is the Euclidean vector inner product
- Good for smaller problems but can quickly become very memory intensive for complex tasks/wider networks

$$g(x, w) = w^T x$$

Types of Deep Networks

2. Convolutional

- Linear function is the discrete convolution of the input and the weights

- This is to make use of what is called the receptive field: the weights are defined on a small convolution kernel and convolved all over the image to produce a feature map (the convolution of the whole image with one kernel) to see what features that specific kernel picks out

- Many kernels can be chosen increasing the number of feature maps

- This greatly speeds up the training time due to weight sharing

- Each proceeding layer picks out more complex features with the kernels

Limitations of Deep Networks

- “Vanilla” deep neural networks can only solve two different classes of problems:

  1. Classification

  2. Regression

- Any other problem that does not fall into these categories will not be solved correctly by deep neural networks

- (solutions may look correct but that doesn’t mean they are!)

I. Solar Image Classification

Deep CNN for Solar Images I

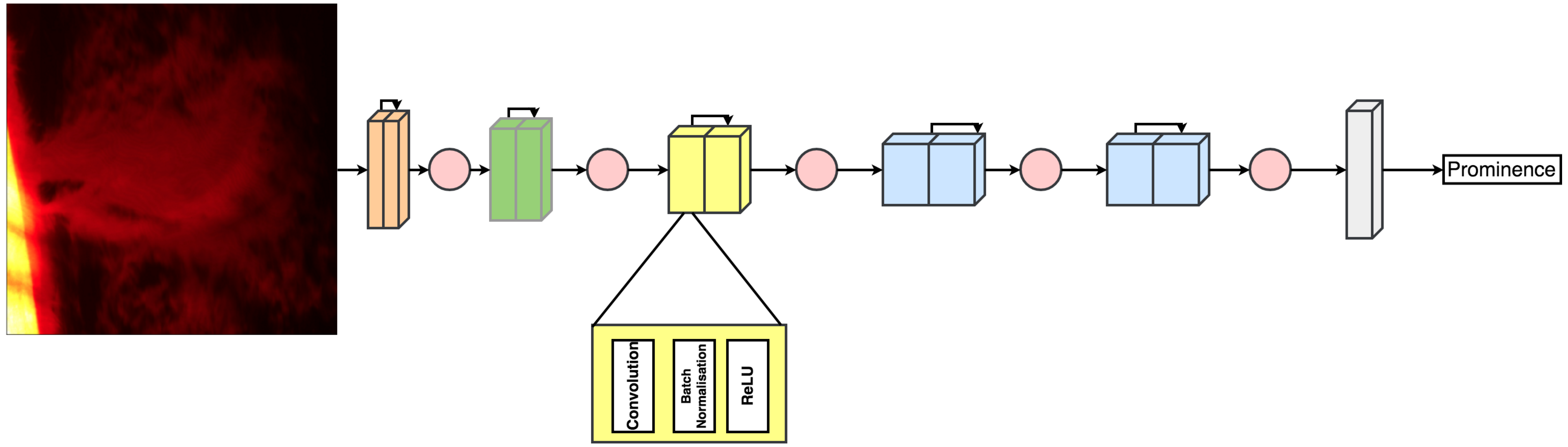
- We are entering the petabyte age of solar data.
- We will need fast, accurate algorithms that can sort images based on image properties.
- Deep CNNs can be trained for extremely fast (subsecond) inference on images once they have learned the geometry of the features we would like it to identify.

Deep CNN for Solar Images II

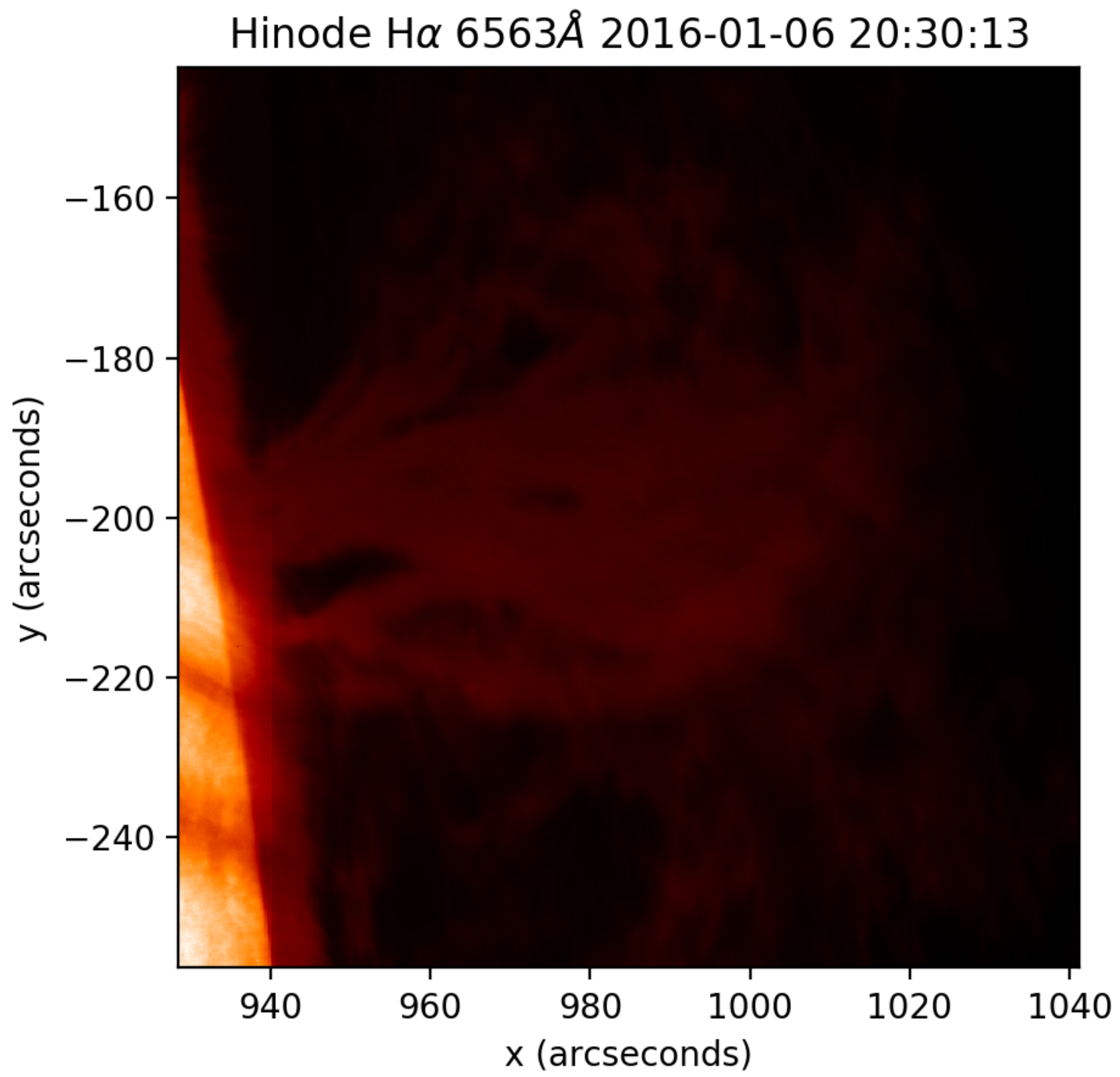
- We have utilised a 13 layer CNN to learn what a subset of solar features look like: prominences, filaments, sunspots, flare ribbons and the quiet Sun.

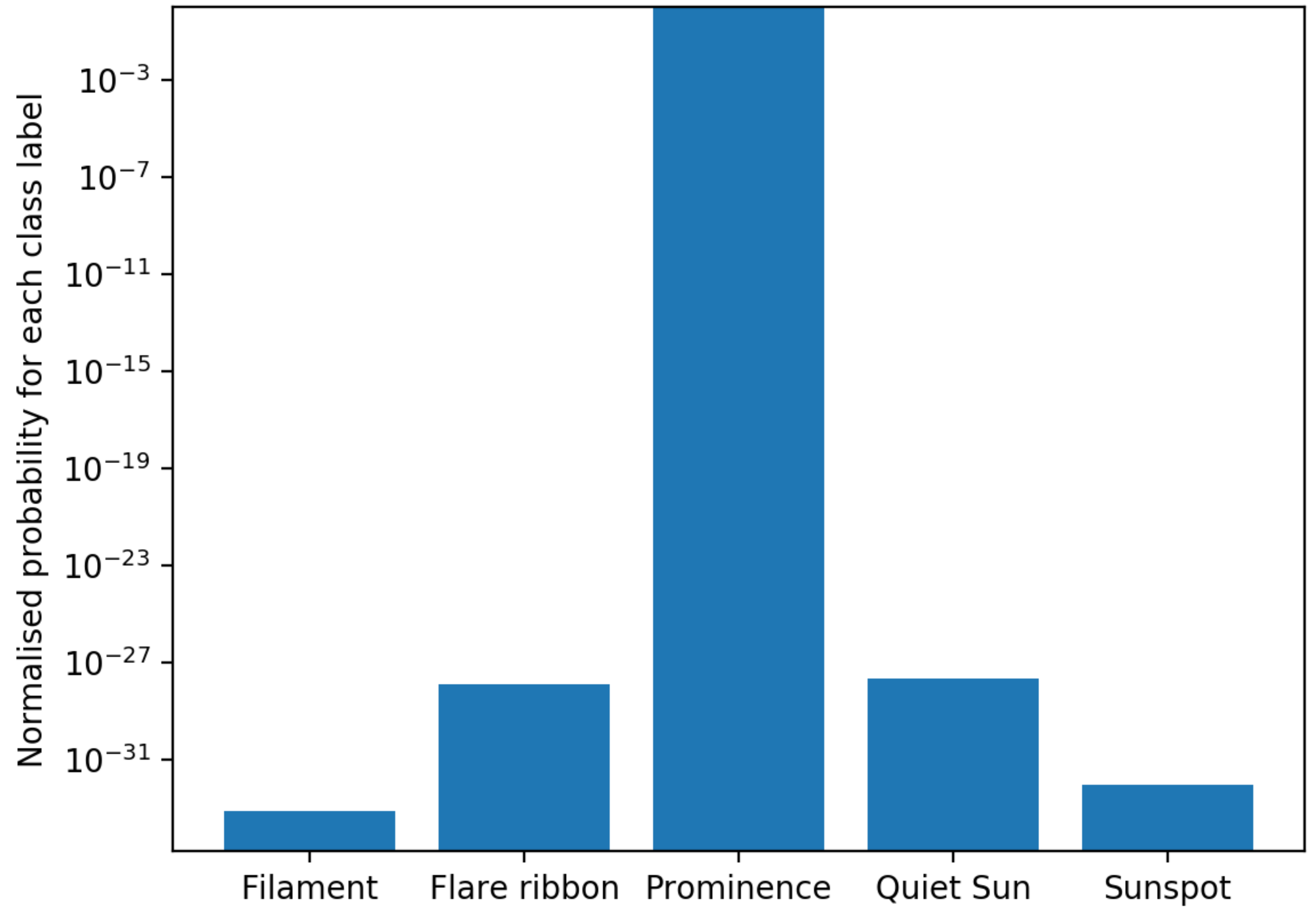
- This was trained using Hα Hinode SOT observations

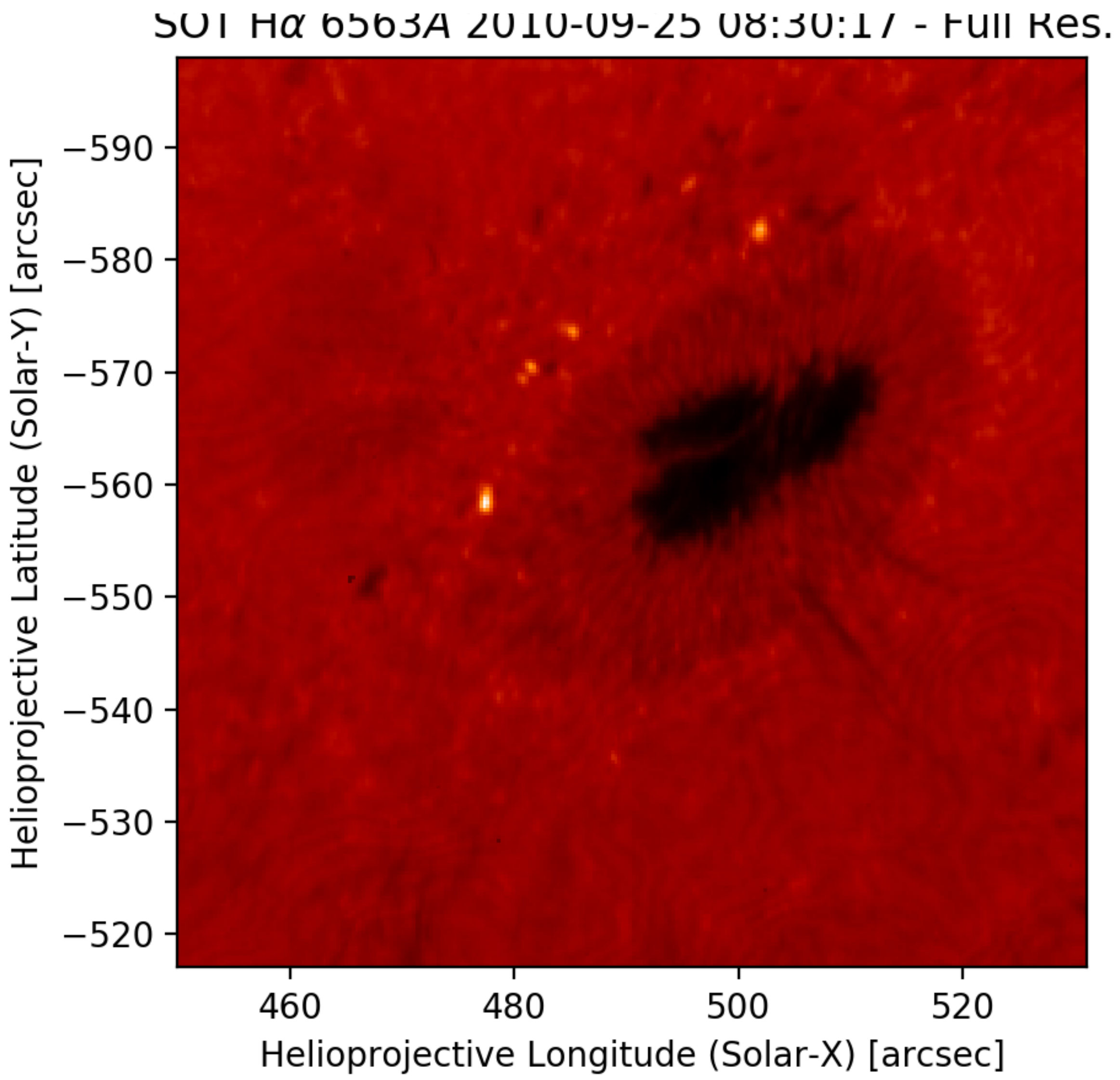
- On GPU, this network can classify 1300+ images in <10s.

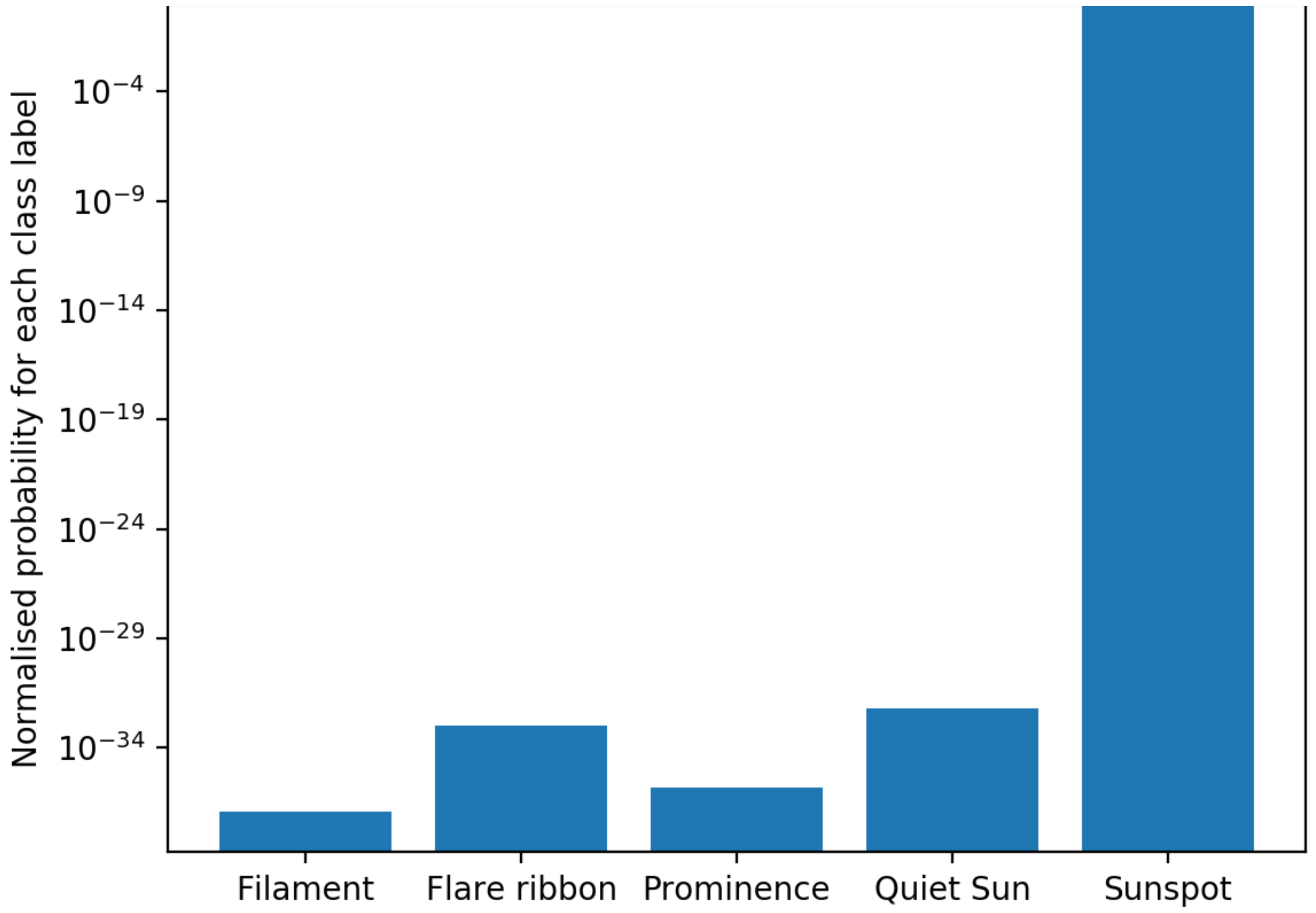


Deep CNN for Solar Images III









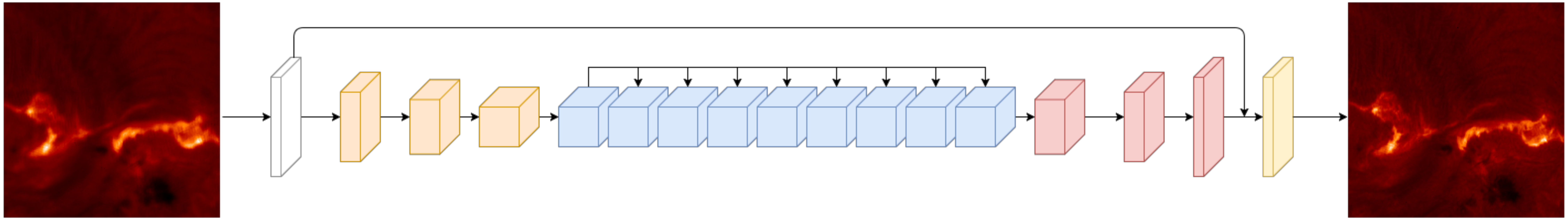
II. Correcting for atmospheric seeing

Correcting for seeing in flare observations I

- Turn to semi-supervised learning for more complex tasks— generative adversarial networks (GANs)
- GAN consists of two deep networks playing a game against each other
- Discriminator (D) and Generator (G)
- D classifies the images are being real or fake
- G tries to generate images to trick D into thinking they're real

Correcting for seeing in flare observations II

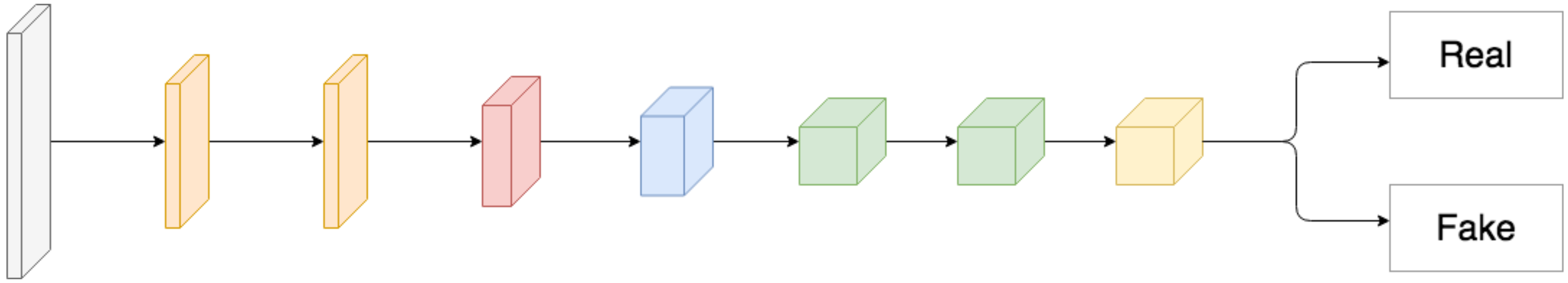
- G is a deep encoder-decoder CNN which learns the mapping from a blurry image to a sharp image



$$\mathscr{L}_X = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} \left(\phi_{i,j}\left(I^S\right)_{x,y} - \phi_{i,j}\left(G\left(I^B\right)\right)_{x,y} \right)^2$$

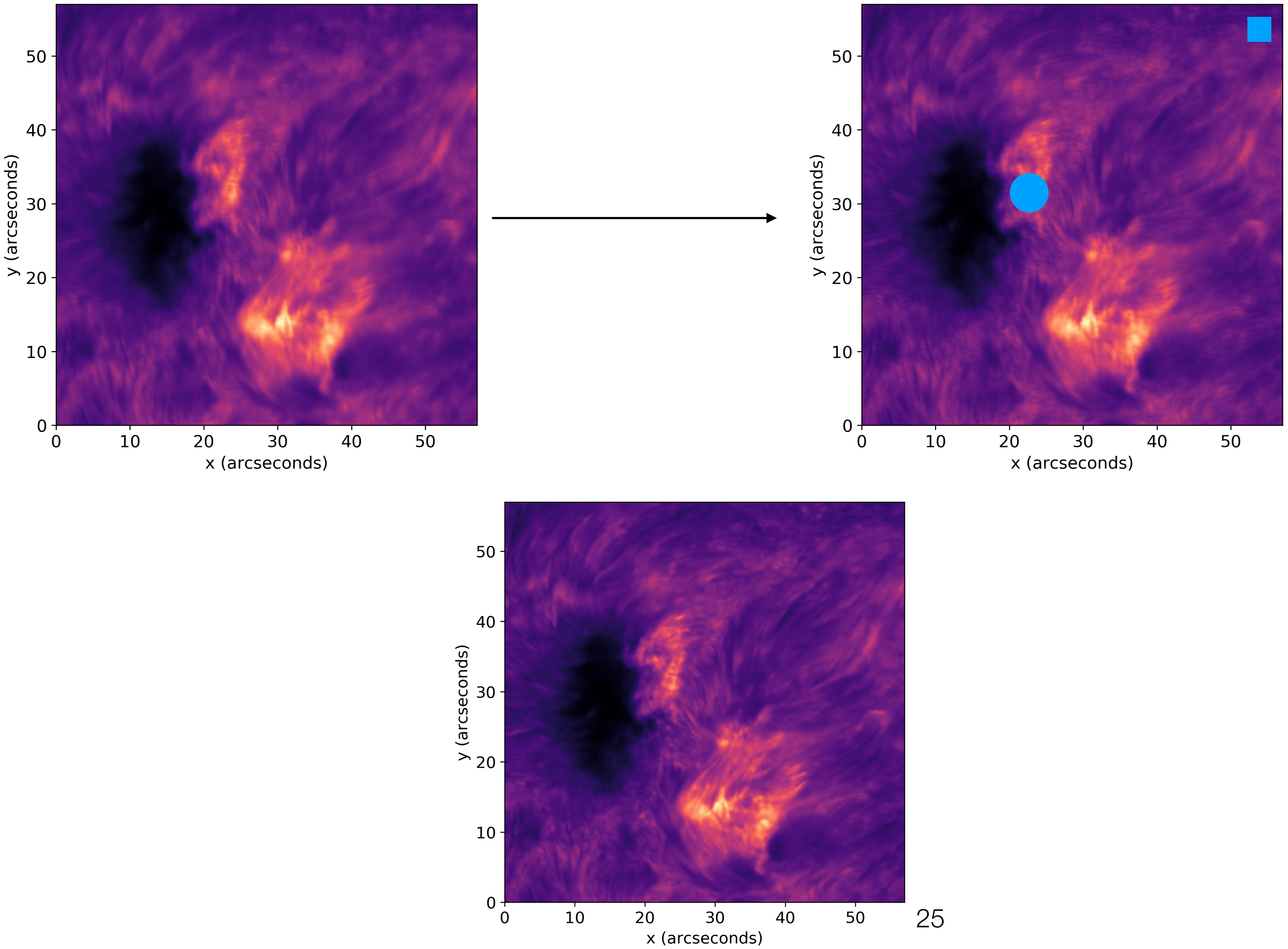
Correcting for seeing in flare observations III

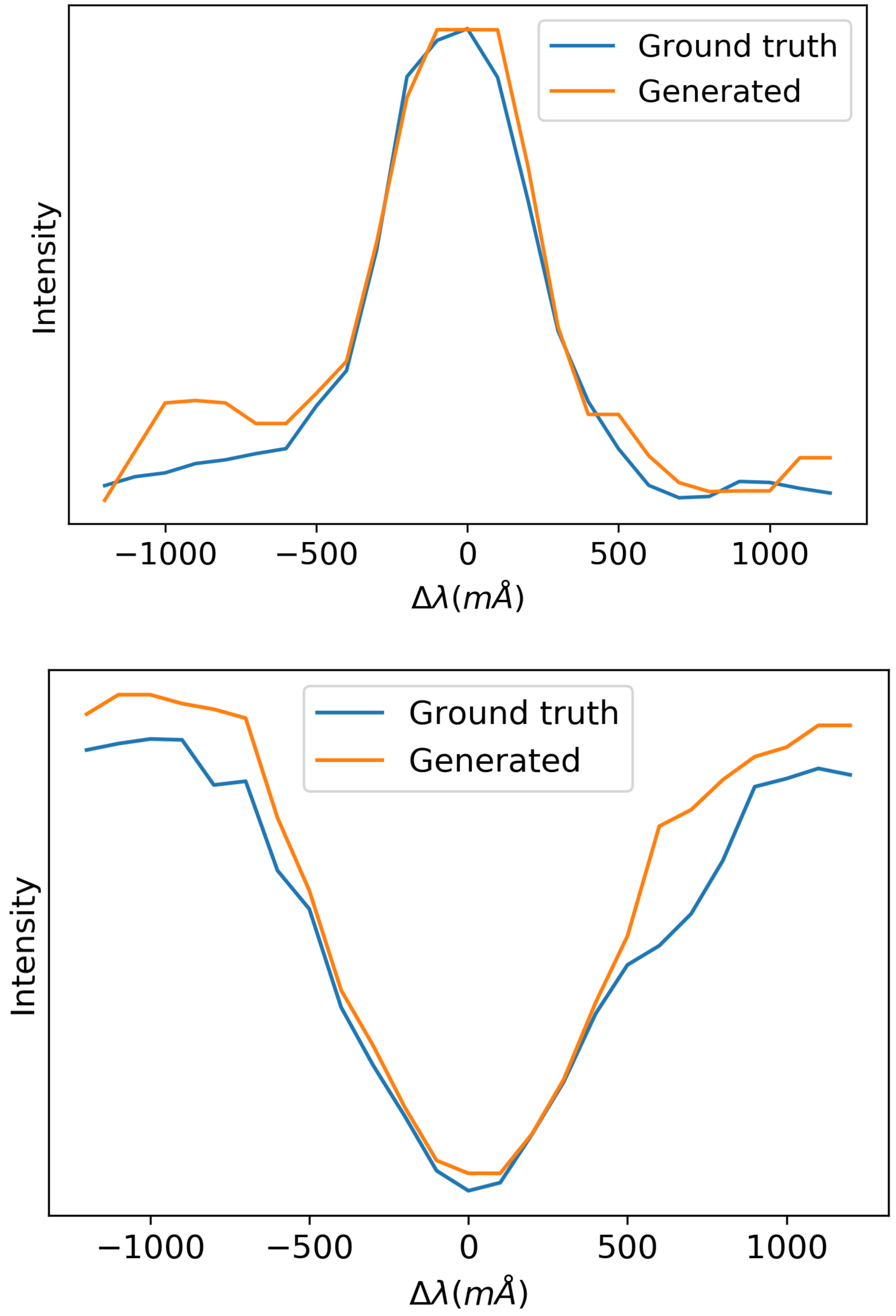
- D is a deep binary classifier CNN which determines whether or not the image passed from G is real or fake



$$\mathscr{L}_{GAN} = \sum_{n=1}^{N} -D\left(G\left(I^{B}\right)\right)$$

Correcting for seeing in flare observations IV

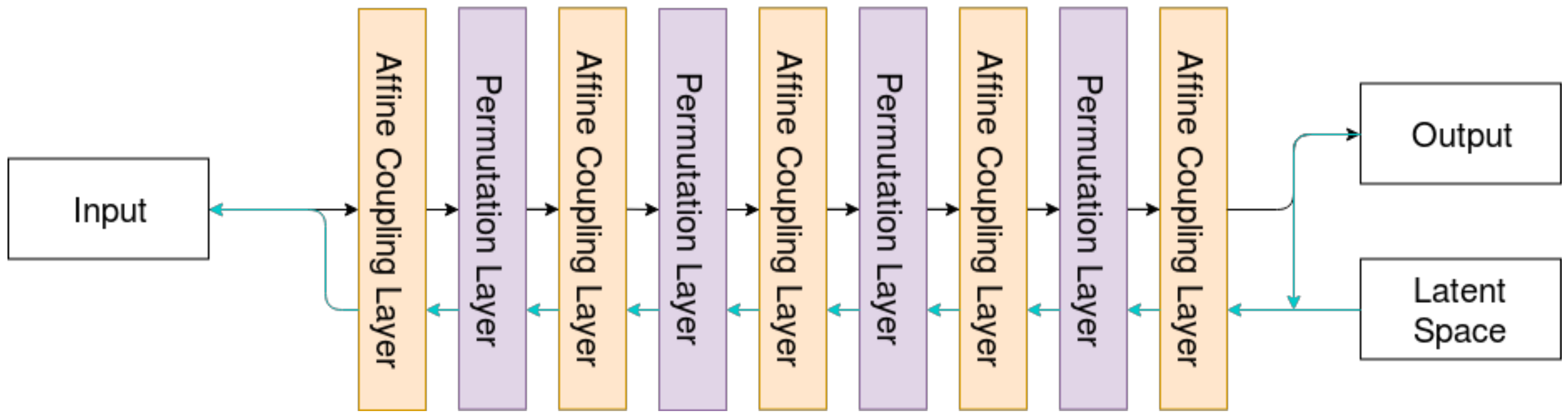




Applications

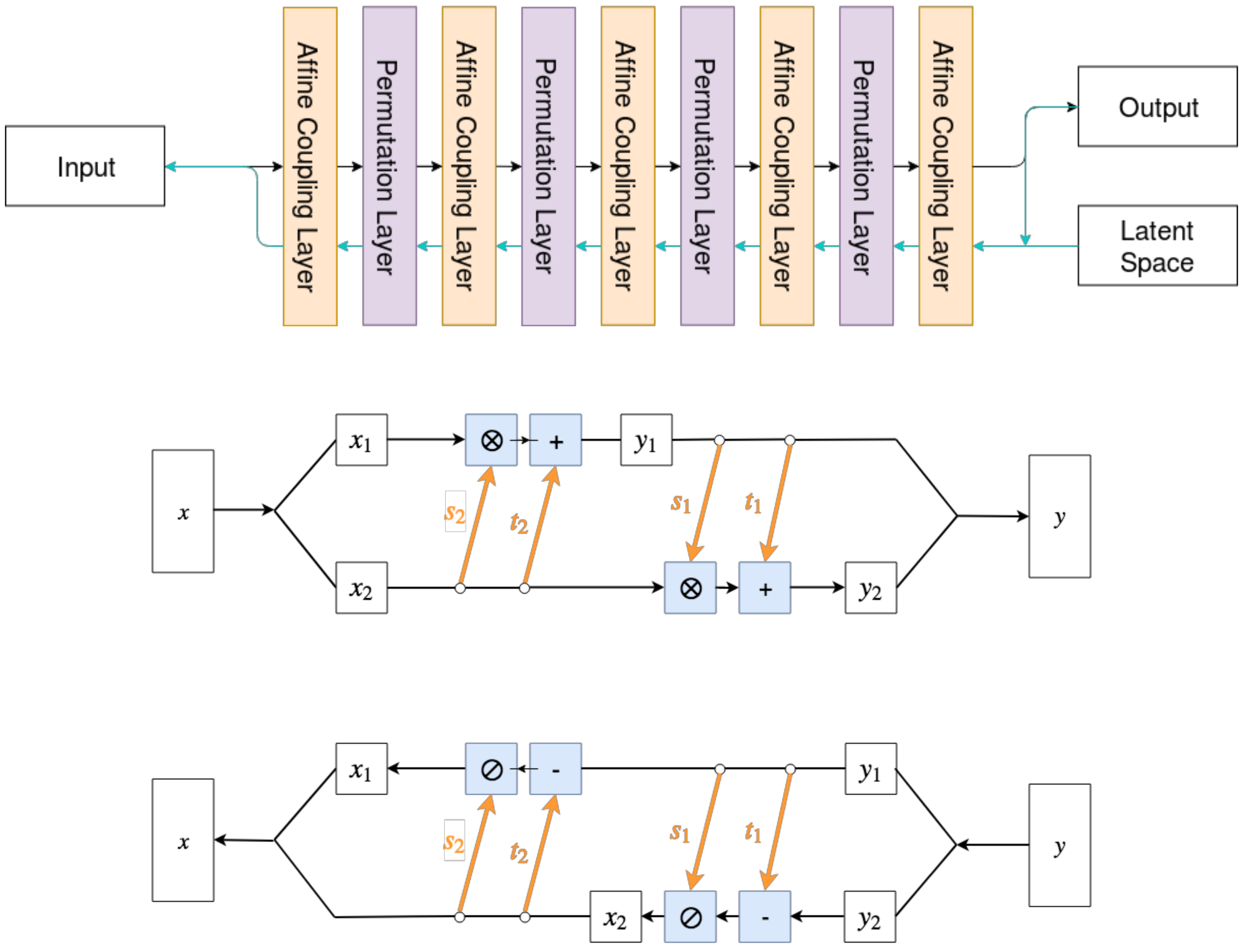
III. Inversion of solar flare line profiles

Inversion of solar flare chromospheric line profiles I



- Due to loss of information, inversions are ambiguous processes
- Traditional deep networks cannot resolve these ambiguities
- We propose using an invertible neural network (INN) to solve the inverse problem

Inversion of solar flare chromospheric line profiles II



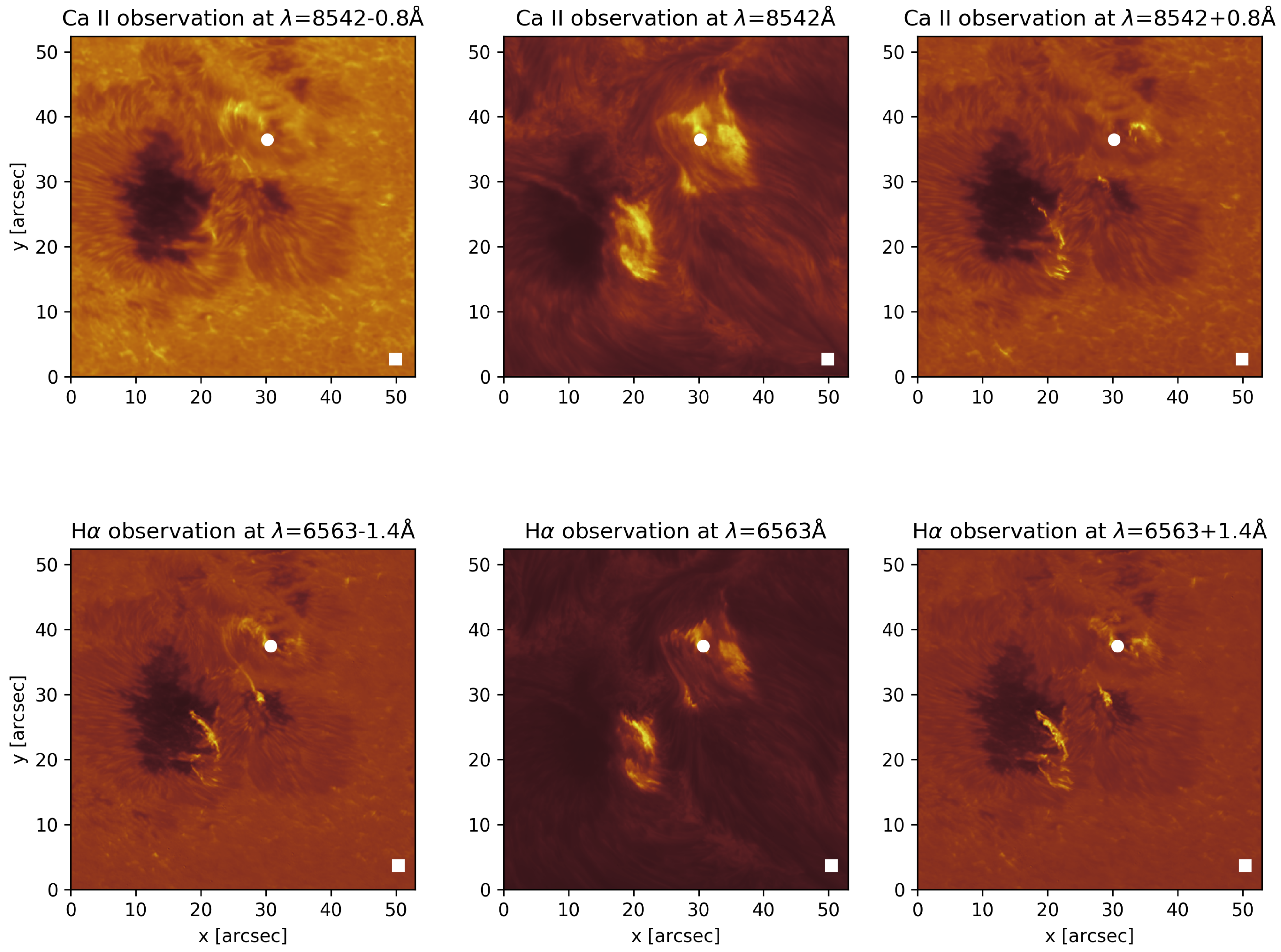
- INN is constructed of affine-coupling layers which are easily invertible
- This means that we can define the bijection from input to output and latent space directly and learn this mapping

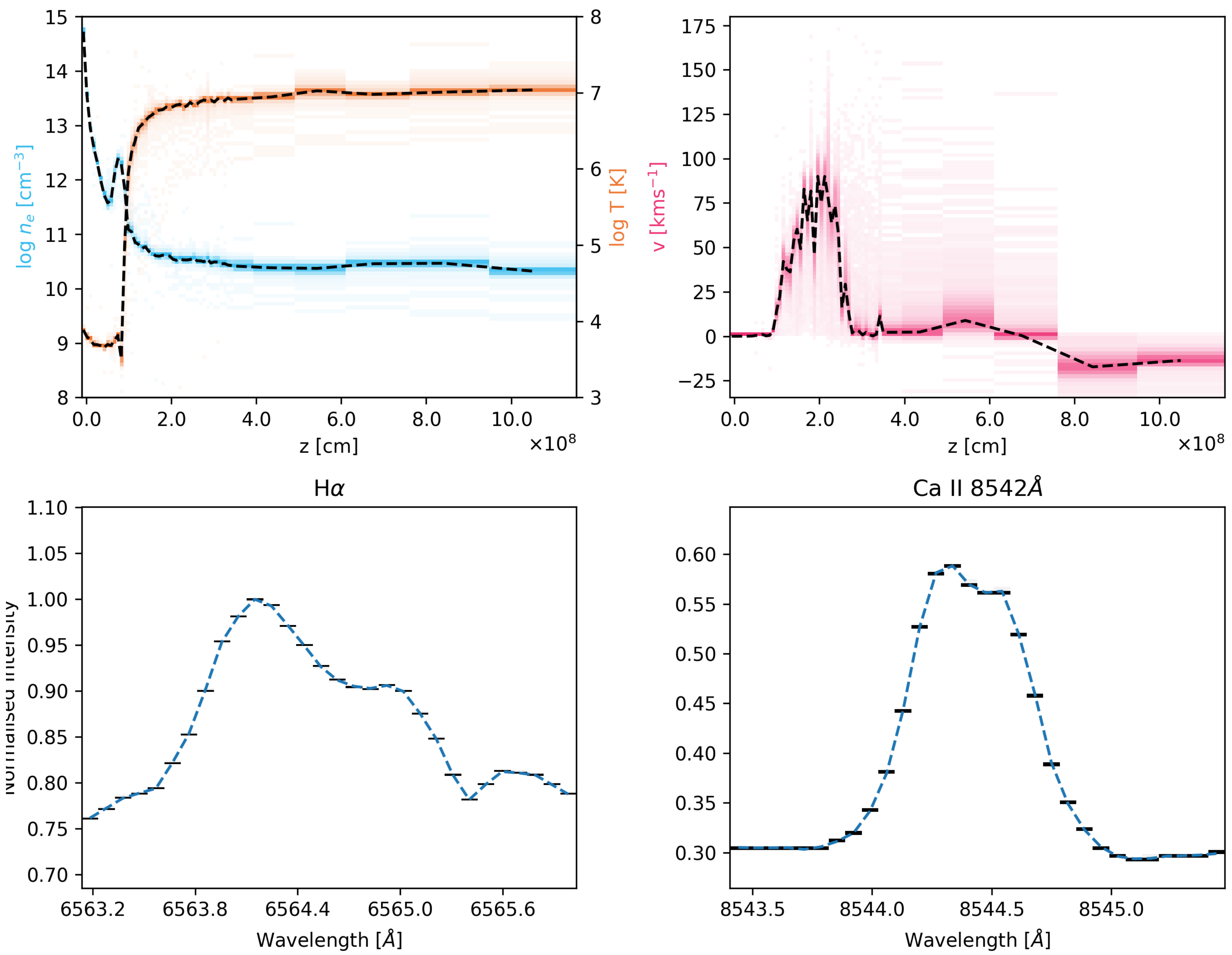
University of Glasgow

Inversion of solar flare chromospheric line profiles III

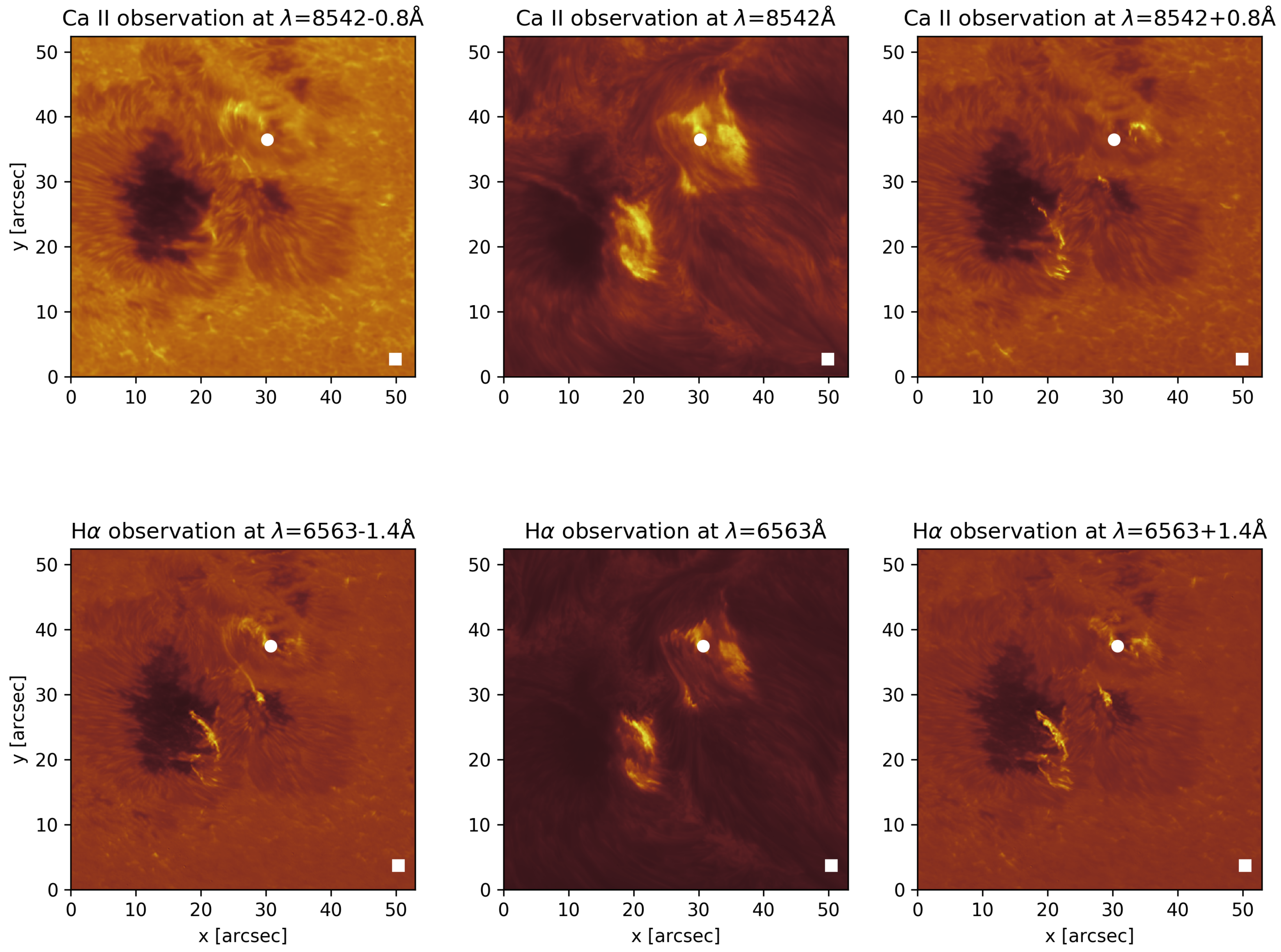
- This is trained on a database of RADYN atmospheres
- Implies that flares follow a RADYN-like atmosphere (BIG assumption, but we don’t have a better forward model)
- Inversions using this architecture can be done ridiculously quickly due to learned model (20,000 inversions takes <1s on GPU)

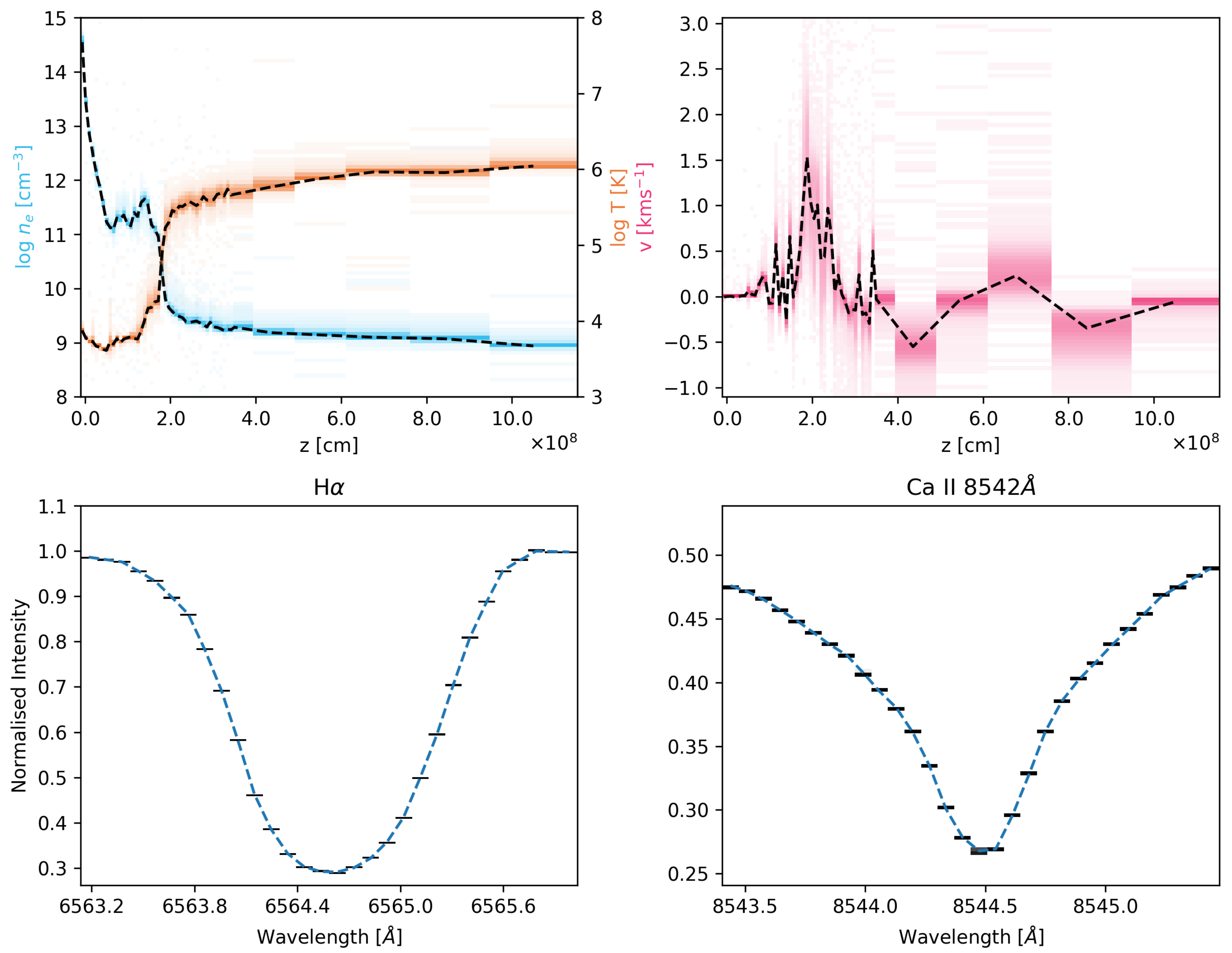
Inversion of solar flare chromospheric line profiles IV





Inversion of solar flare chromospheric line profiles V





Where do we go from here?

- Perform inversions on whole images of flares
- Combine with contribution functions to produce Dopplergrams of multiple flares
- Data pipeline: correcting for seeing, ribbon tracking, inversions