DR ADRIAN BEVAN

MACHINE LEARNING IN HIGH ENERGY PHYSICS

5) DEEP LEARNING

Lectures given at the department of Physics at CINVESTAV, Instituto Politécnico Nacional, Mexico City
28th August - 3rd Sept 2018

LECTURE PLAN

- Introduction
- Deep MLPs
- Convolutional neural networks
- Adversarial networks
- Using auto-encoders
- Examples
- Summary
- Suggested reading

INTRODUCTION

- Deep networks have had a lot of media attention in recent years and are possible because of advances in computing technology - specifically multi-core CPUs and GPUs.
- They are computationally very expensive and require large samples of data and a lot of training epochs to ensure convergence to some optimal solution.
- We can adapt the algorithms we have discussed so far to turn those neural networks into deep networks.
- There are new forms of network that we will also discuss.

DEEP MLPs

- What is a deep network?
  - Different people use different definitions for what a deep network is.
  - An MLP with more than a few layers is called a deep network.
- Configurations vary from a few very wide layers (hundreds of nodes) to a large number of narrow layers (with similar to the dimensionality of the input feature space); as well as everything in-between those two extremes.

DEEP MLPs

- General remarks about using deep MLP's:
  - Typically need a large training sample to evaluate the HPs for a deep network.
  - Dropout is used to mitigate overtraining, batch learning to accelerate optimisation.
  - Deep networks with sufficient training data are able to learn the underlying patterns from lower level features (e.g. four-vectors) that would normally be present in higher level derived features (e.g. invariant mass, missing mass, etc.).
  - More care and attention to the training process is required for deep networks than for single layer or MLPs with small numbers of layers.
  - Computing resource required for training increases significantly with complexity (number of HPs).

CONVOLUTIONAL NEURAL NETWORKS (CNNS)

- Input data
- Convolution layers
  - Padding
- Pooling
  - Max-pooling and average pooling
- Convolutional Neural Network (CNN) architectures
  - Revisit input data

CNNs: INPUT DATA

- CNNs take advantage of spatial correlations of the input feature space.[1]
- This is typically in the form of image data.
  - Each pixel corresponds to a feature for each colour that is encoded in it.
  - Greyscale images have a depth of 1, and so the dimensionality of the feature-space is n_{pixels} x m_{pixels}.[1]
  - Colour images have a depth of 3 (R, G, B); so the dimensionality of the feature space is 3 x n_{pixels} x m_{pixels}.[1]

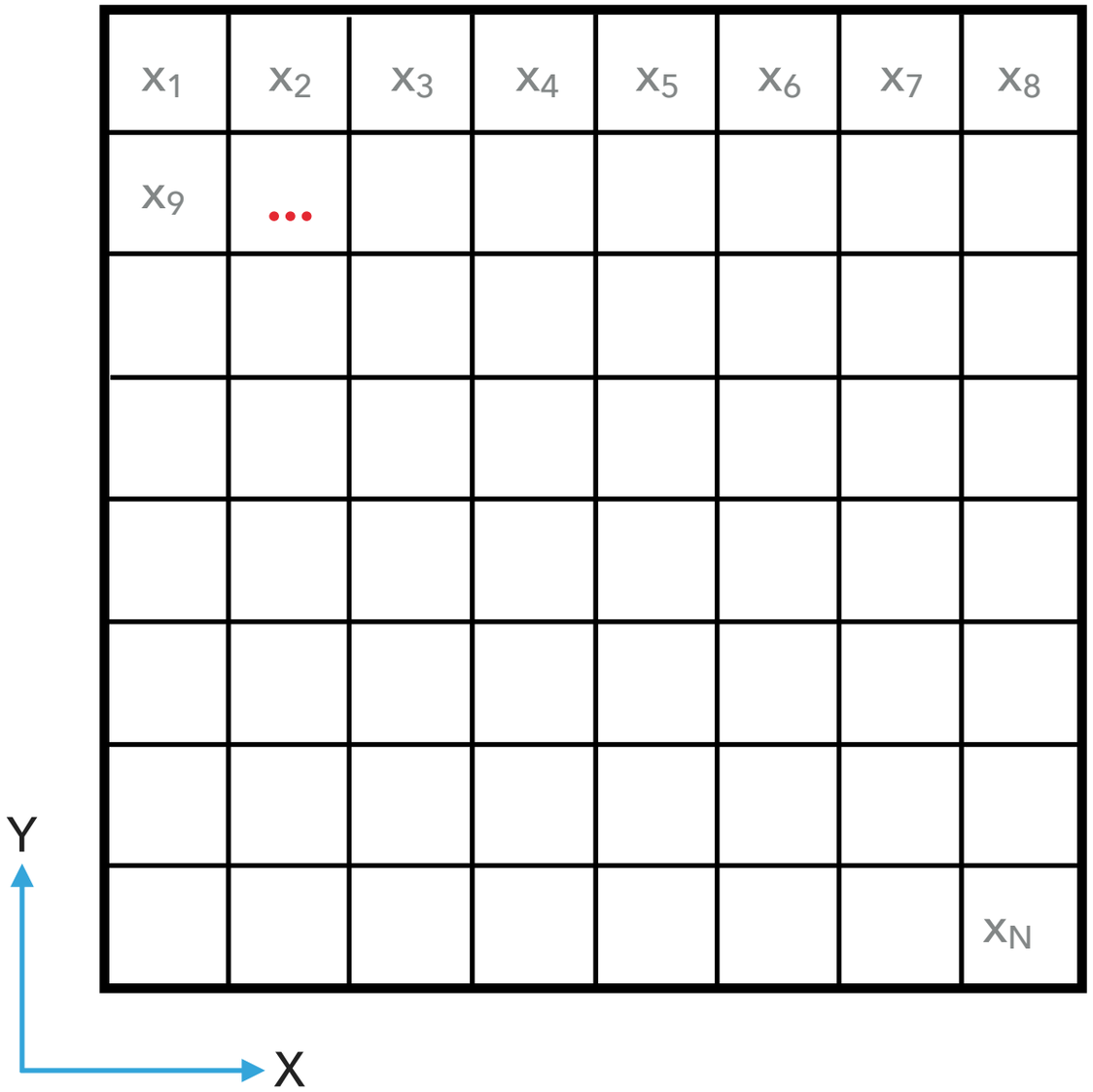
[1] K Fukushima, Bio. Cybernetics **36** p193-202, 1980.
[1]Typically CNNs are applied to square images.

CNNs: INPUT DATA

- Some analyses take pairs of variables and create 2D histograms from those (e.g. p_T and η); normalise the content to lie in the range [0, 1] & feed those into a CNN.
- Different pairs of features correspond to different "colour channel" layers of the CNN input image.

[1] K Fukushima, Bio. Cybernetics **36** p193-202, 1980.

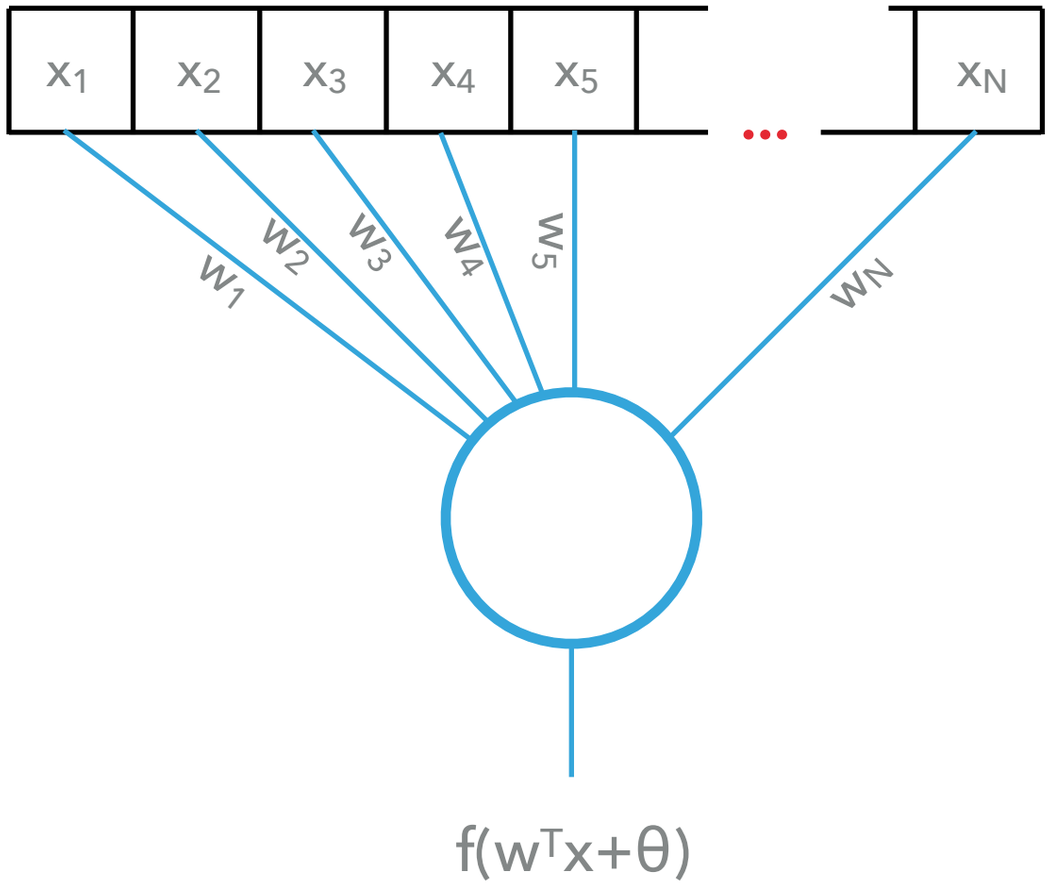
CNNs: INPUT DATA



This is an 8 by 8 array of pixels, that corresponds to a 64 dimensional feature space.

- An MLP can be used to process this data, but you loose the spatial correlations between information in the image.
- The image can be represented by a a line of features.
- Doing this removes the spatial correlations and would naturally lend itself to being processing by a perceptron; i.e. $f(w^T x+\theta)$.

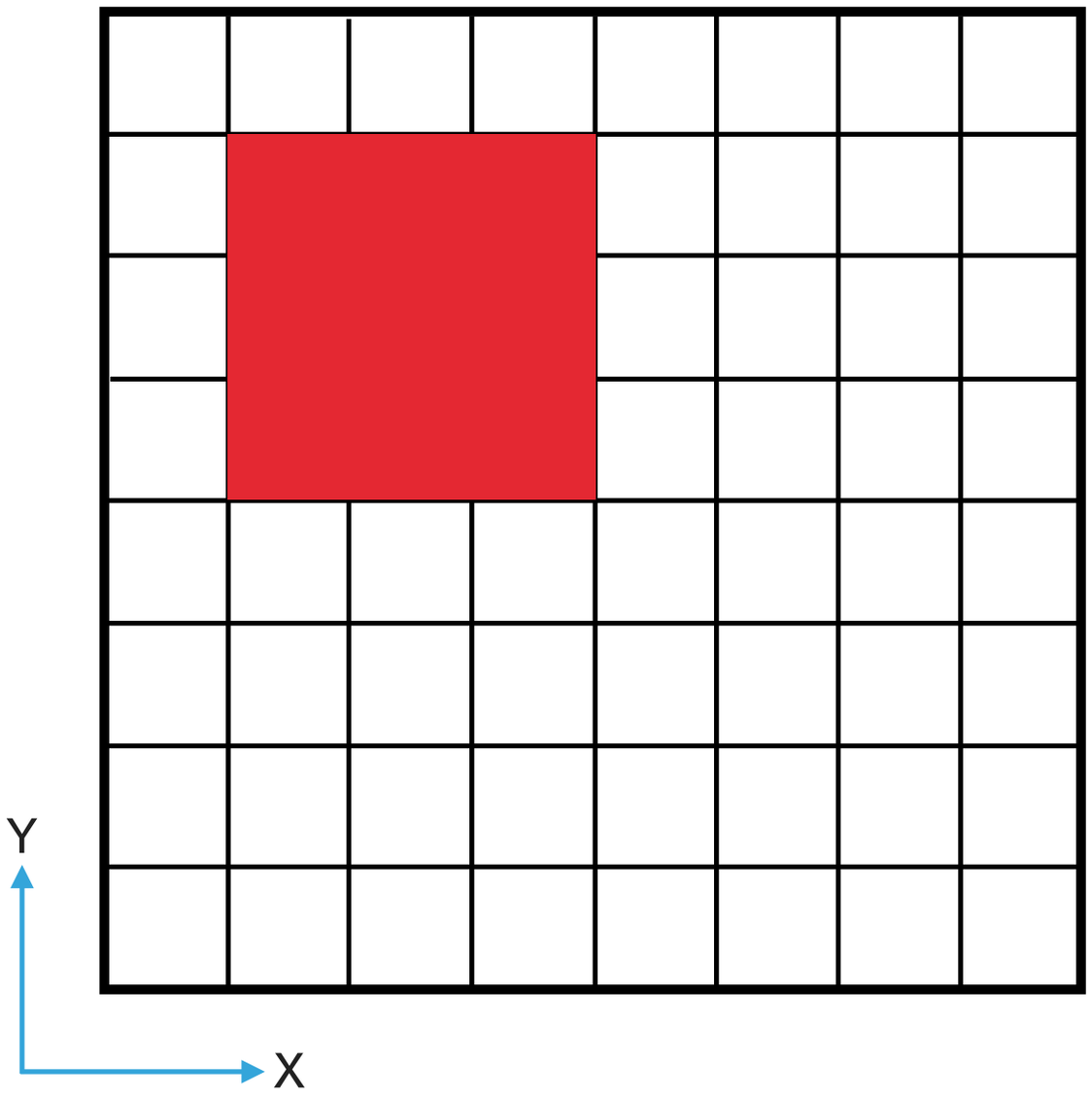
CNNs: INPUT DATA



This is an 8 by 8 array of pixels, that corresponds to a 64 dimensional feature space.

- An MLP can be used to process this data, but you loose the spatial correlations between information in the image.
- The image can be represented by a a line of features.
- But doing this removes the spatial correlations and would naturally lend itself to being processing by a perceptron; i.e. $f(w^T x+\theta)$.

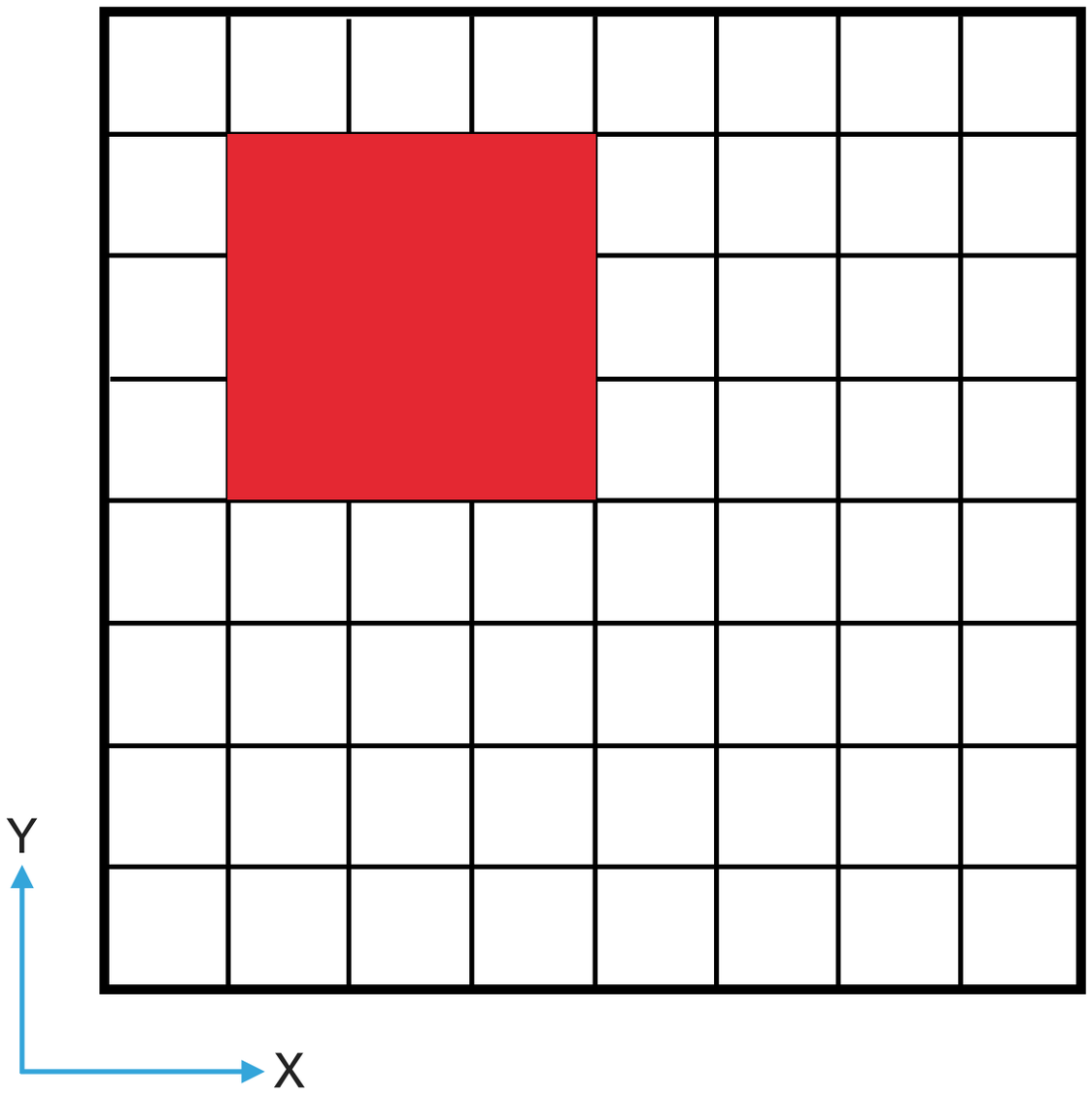
CNNs: CONVOLUTION LAYERS



This is an 8 by 8 array of pixels, that corresponds to a 64 dimensional feature space.

- We can split the image up into a smaller grid of pixels (filter), and search for a pattern in that grid.
  - In this example we take a 3x3 grid of pixels.
  - We can compute a numerical convolution of these 9 pixels using $f(w^T x+\theta)$.
- Spatial correlations within this grid of pixels are used when computing the numerical convolution.
- Larger filters can be used; where odd numbers of pixels are normally used:
  - 1x1: identity transformation preserve the input image;
  - 3x3, 5x5, … ; compute convolution image.

CNNs: CONVOLUTION LAYERS

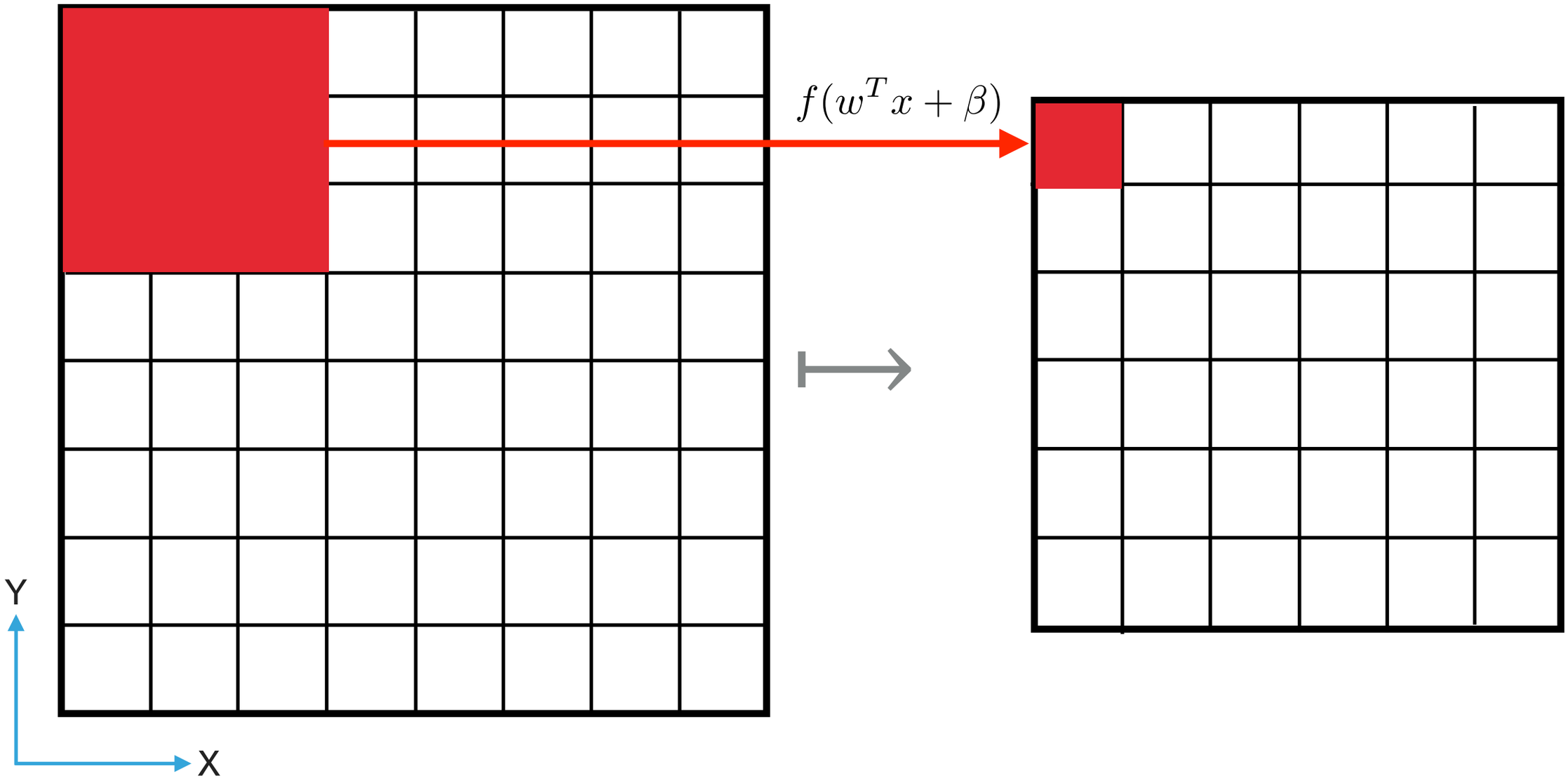


This is an 8 by 8 array of pixels, that corresponds to a 64 dimensional feature space.

- That same "convolution filter" can be used iteratively over the whole input image.
- The output values for each iteration are just the value of the output of a perceptron.
- The set of outputs from running the convolution filter across the input forms a new "convolution image".

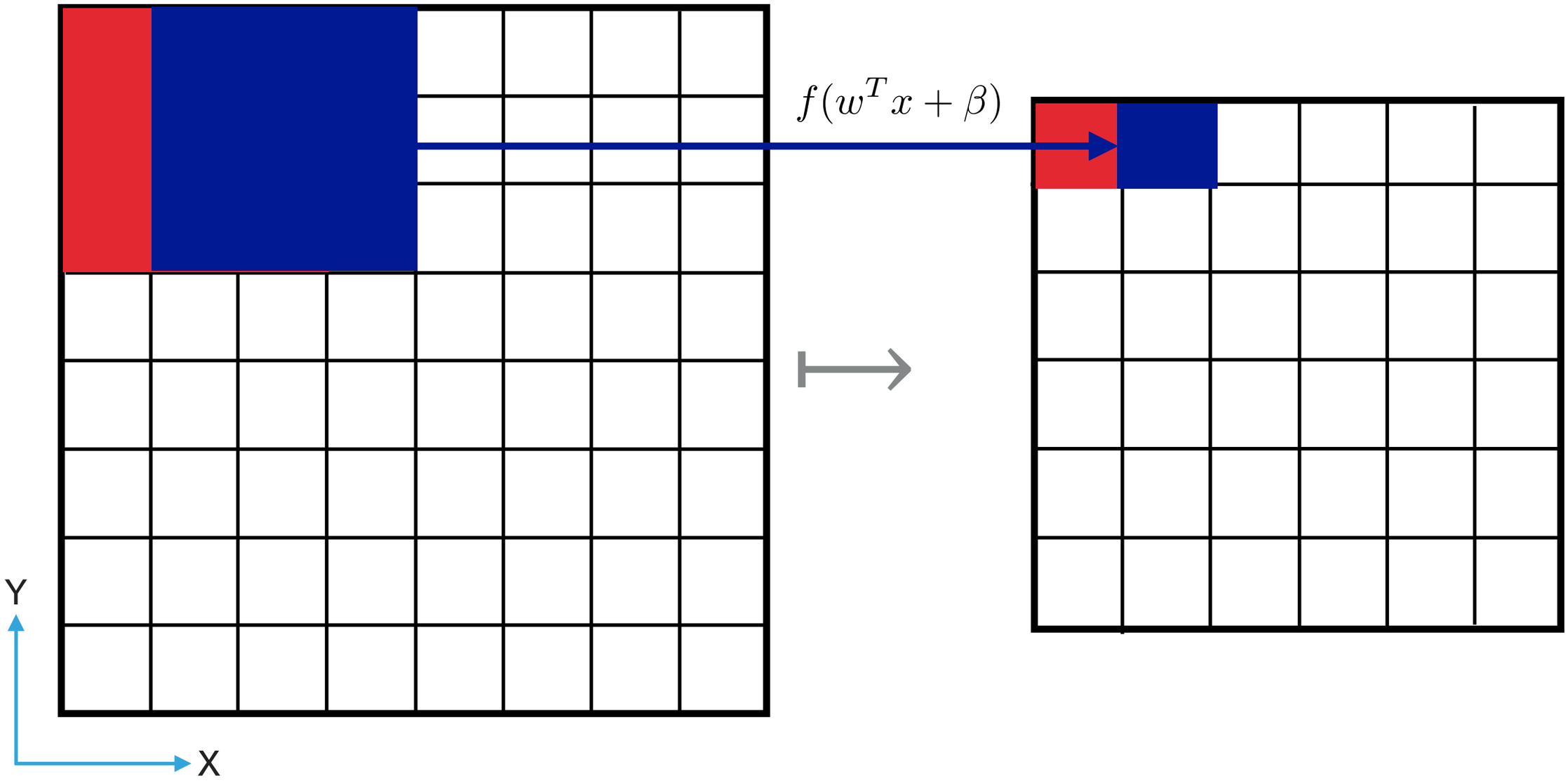
CNNs: CONVOLUTION LAYERS

- If you use an M x M filter on an N x N image, the convolved image is smaller than the original.



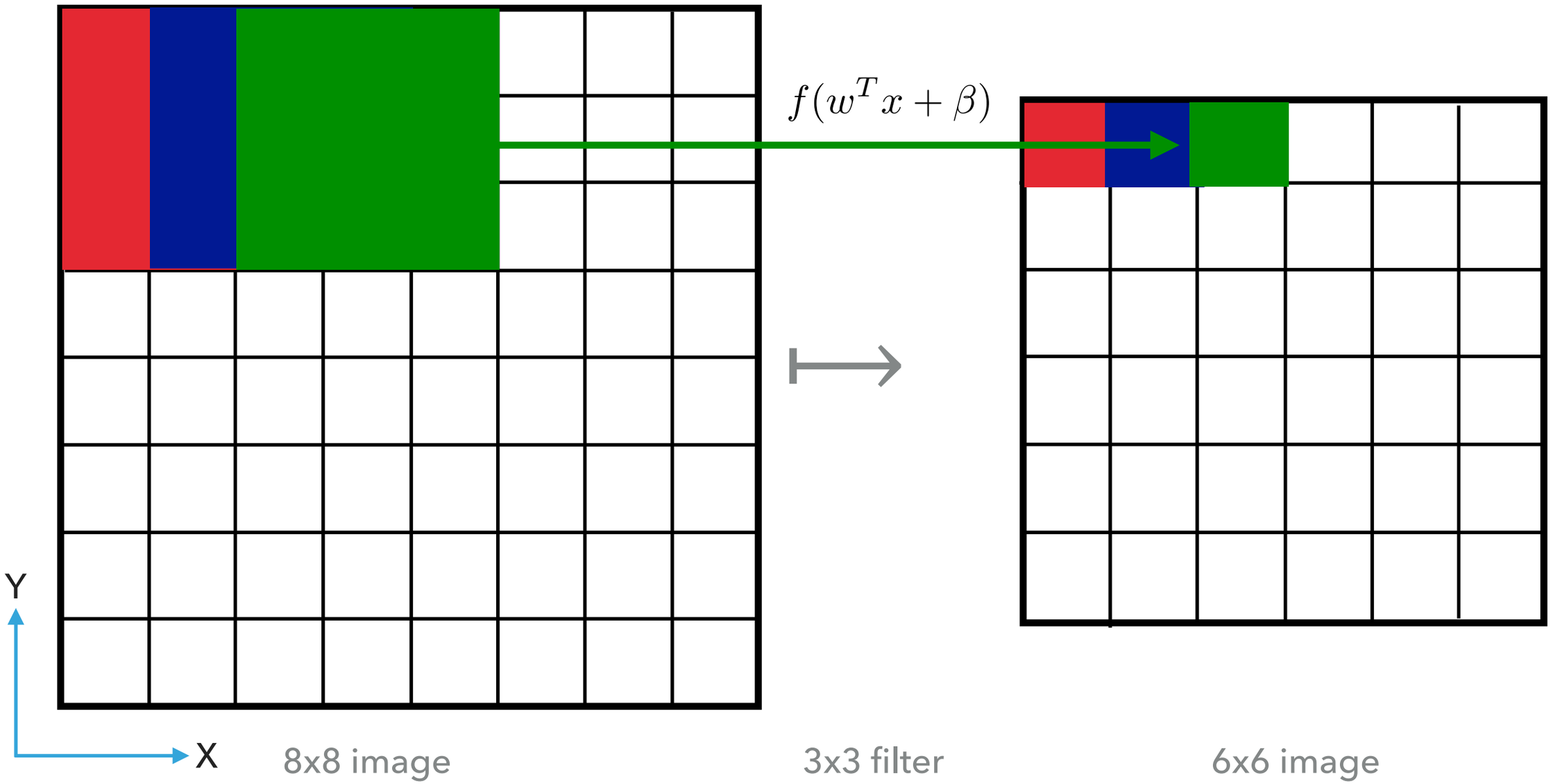
CNNs: CONVOLUTION LAYERS

- If you use an M x M filter on an N x N image, the convolved image is smaller than the original.



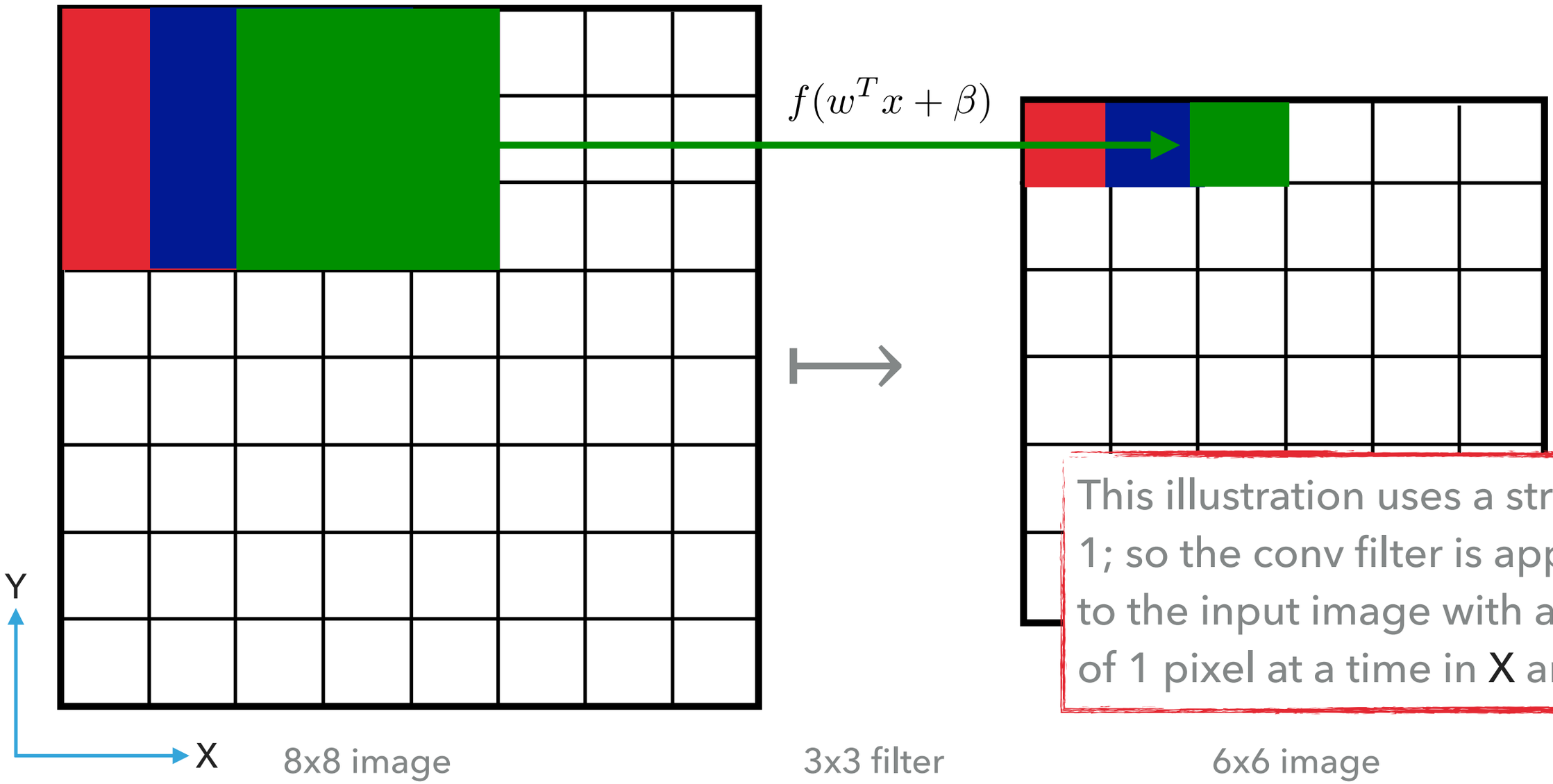
CNNs: CONVOLUTION LAYERS

- If you use an M x M filter on an N x N image, the convolved image is smaller than the original.



CNNs: CONVOLUTION LAYERS

- If you use an M x M filter on an N x N image, the convolved image is smaller than the original.



This illustration uses a stride of 1; so the conv filter is applied to the input image with a shift of 1 pixel at a time in X and Y.

CNNs: CONVOLUTION LAYERS

- (M-1)/2 pixels are lost from the border of the input image in order to create the convolution image.

Image Size	Filter Size	Convolution image size
8x8	3x3	6x6
8x8	5x5	4x4
8x8	7x7	2x2
10x10	3x3	8x8
10x10	5x5	6x6
10x10	7x7	4x4
10x10	9x9	2x2

This illustration uses a stride of 1

- An NxN image becomes a (N-M+1) x (N-M+1) image*.
- Repeatedly convoluting the image reduces the dimensionality of the feature space; which can be undesirable.

*The border is both sides of the image so you loose (M-1)/2 pixels twice; once for each side.

CNNs: CONVOLUTION LAYERS: PADDING

- The dimensional reduction of feature space can be mitigated by padding the original image with a border of width (M-1)/2 pixels.
- The values of the border padding are set to zero (no information provided to the convolution layer).
- Now the original image can be convoluted any number of times (within resource limitations) ***without reducing the dimensionality of the input feature space that contains non-trivial information***.

CNNs: CONVOLUTION LAYERS

- Each convolution filter is tuned to identify a given shape when scanning through an image.
  - These can be edge, line or other shape filters.
- By using a set of convolution filters, one can pick out a set of different features in an image.
- The weights for these filters are usually initialised randomly using a truncated Gaussian distribution with output >0.
  - This is to avoid negative weights.

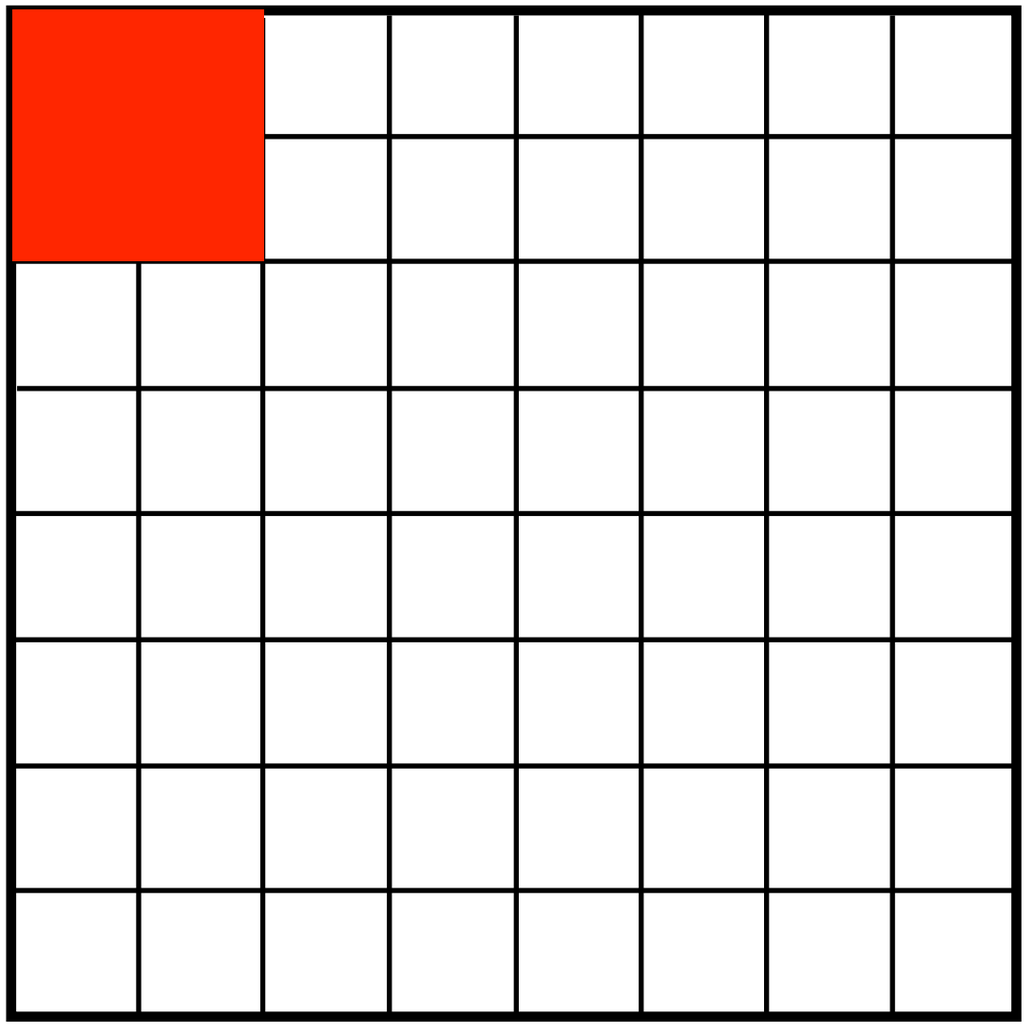
CNNs: POOLING

- The dimensionality of the features in a convolutional layer is large; numerically it is convenient to reduce the dimensionality for further processing.
  - High resolution images are not required to be able to identify shapes of objects;
  - Can make a lower resolution representation and still reach the same conclusion.
  - Pooling is a mechanism that allows you to achieve this.[1]
- Define a filter size for pooling (e.g. 2x2) and then perform an operation on the pixels to compute:
  - Maximum value (max pooling): useful to suppress noise when information is sparse and the number of pixels having a significant value is expected to be low.
  - Average value (average pooling): Averaging pixels values can give a smaller variance on the information contained in those pixels.
- Ref. [1] provides an analysis of these two approaches.

[1] Boureau, Y. L., Ponce, J., & Lecun, Y. (2010). A theoretical analysis of feature pooling in visual recognition. In ICML 2010 - Proc., 27th Int. Conf. on Machine Learning (pp. 111-118)

CNNs: POOLING

- The pooling process is applied to each individual part of the image (i.e. dimensional reduction)



Average pooling is just applying the following to each set of pixels.

$$f = \frac{1}{N}\sum_{i=1}^{N} x_i$$

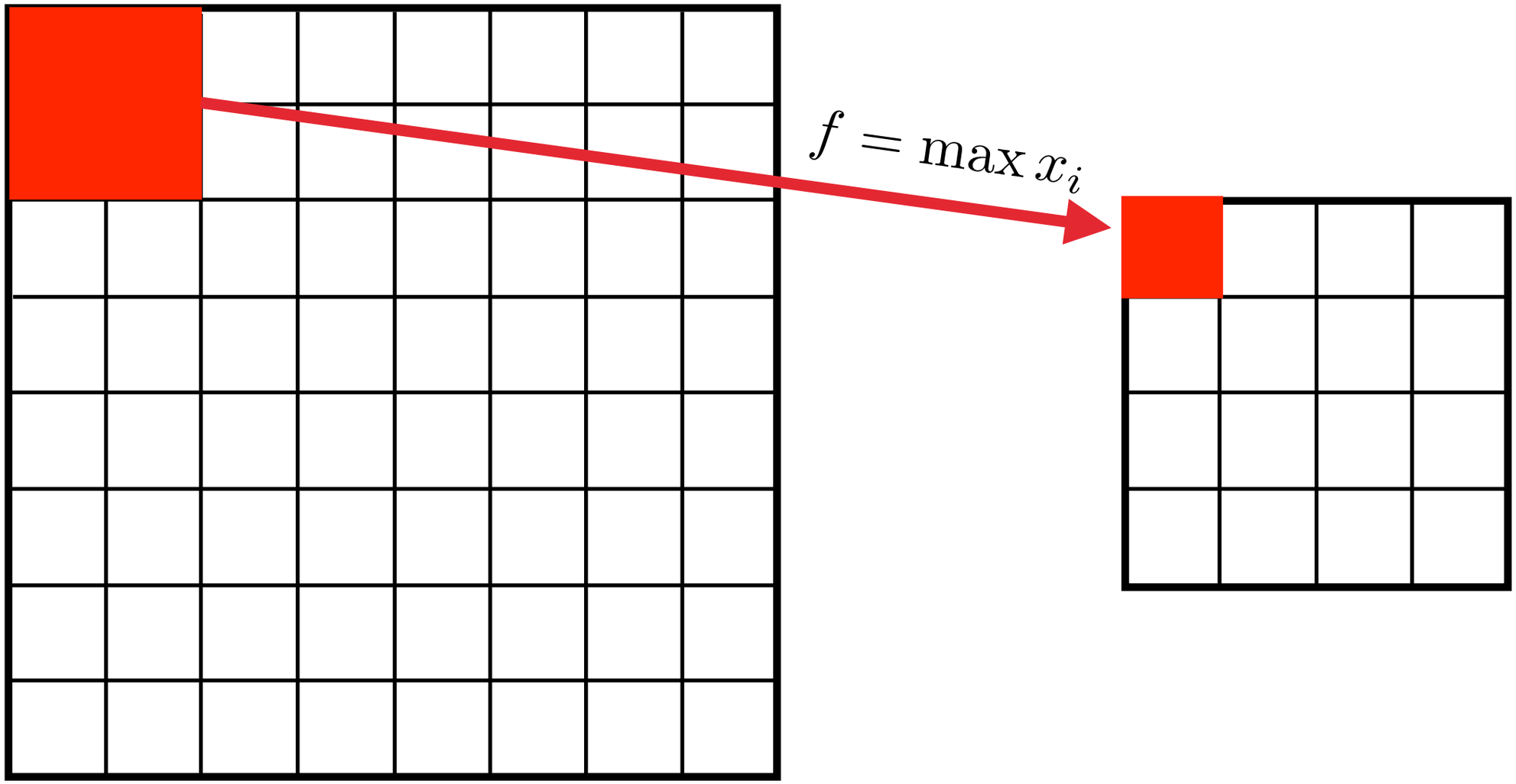
Max pooling is equivalent but taking

$$f = max(x_i)$$

The output is a smaller image.

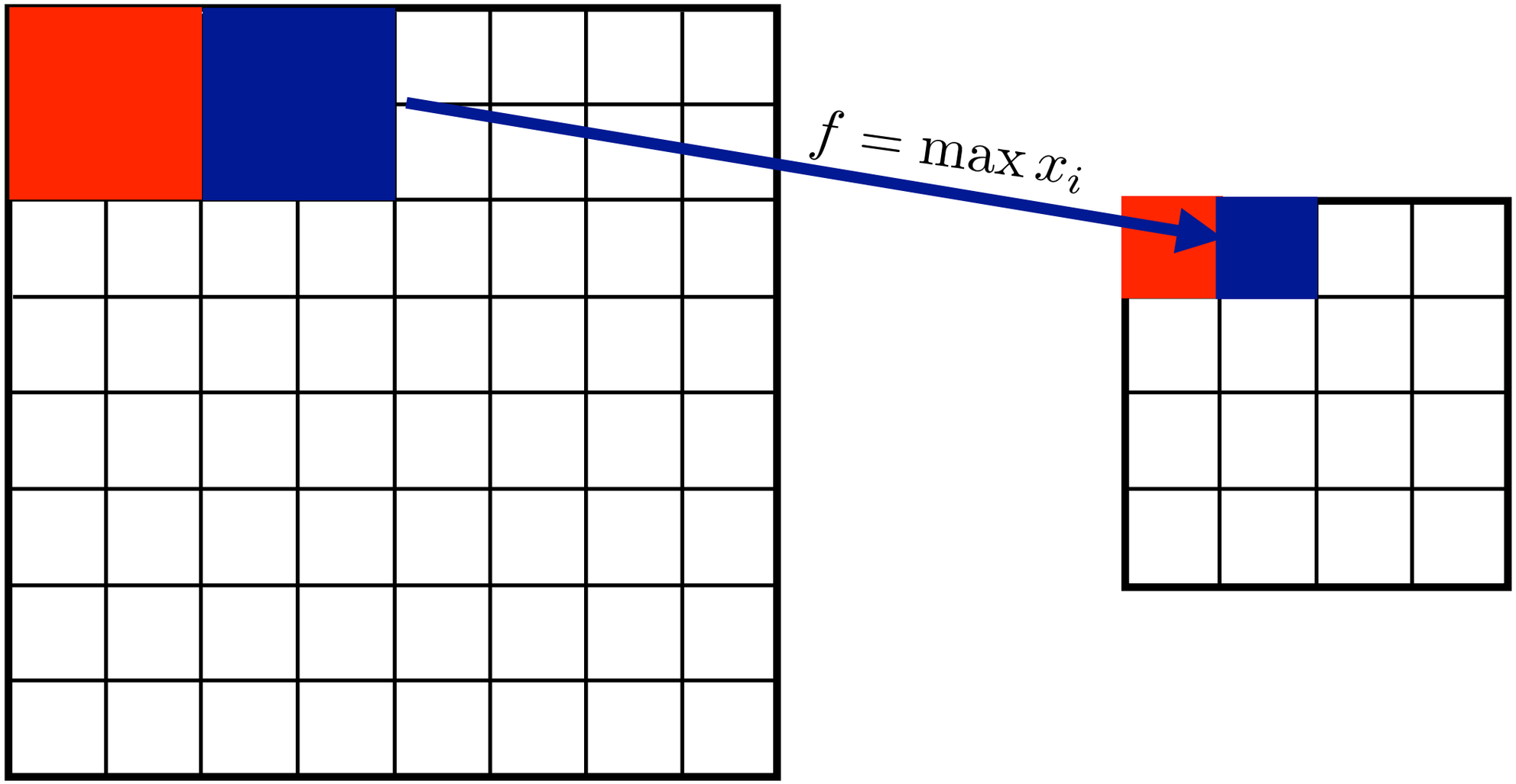
CNNs: POOLING

- The pooling process is applied to each individual part of the image (i.e. dimensional reduction)



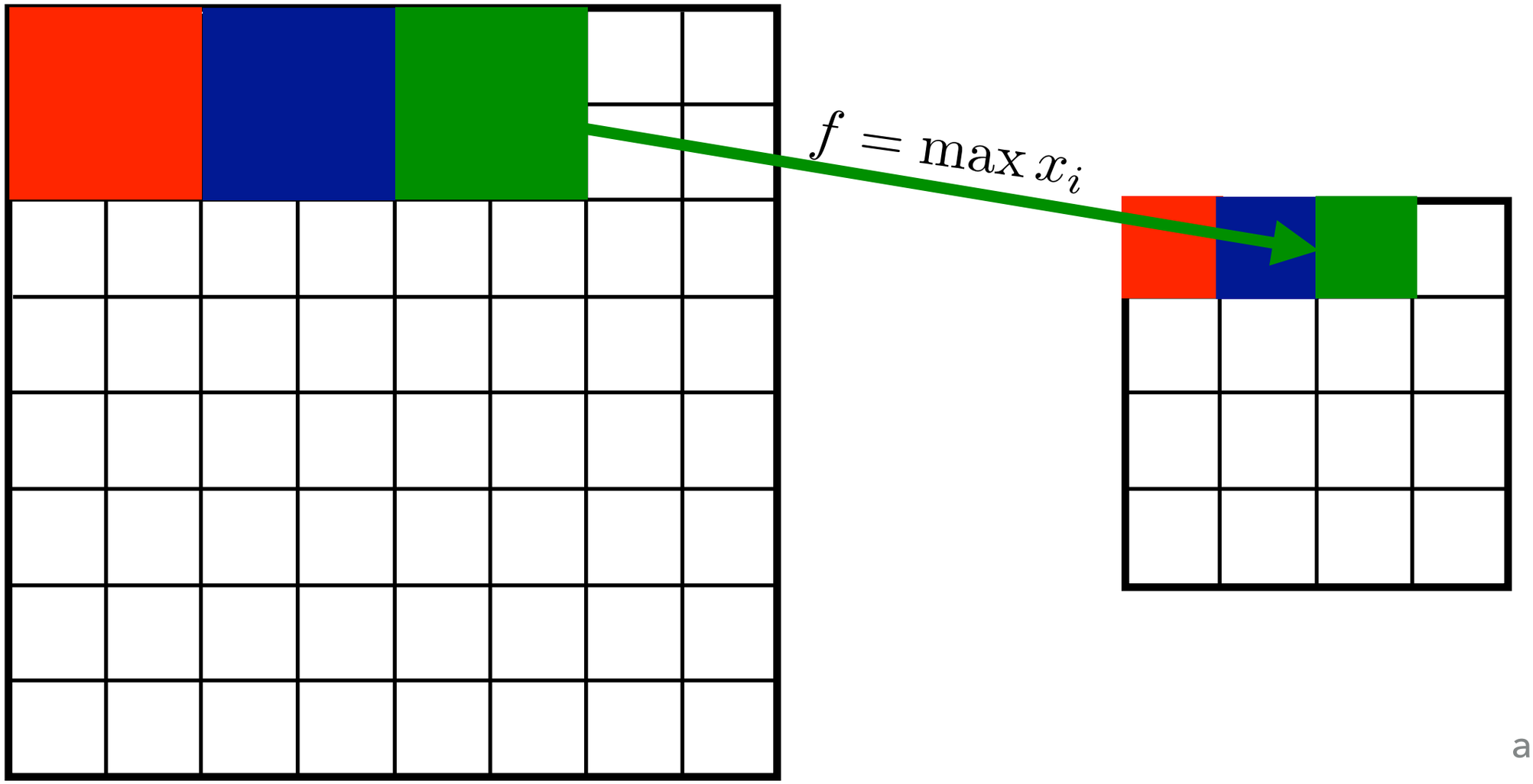
CNNs: POOLING

- The pooling process is applied to each individual part of the image (i.e. dimensional reduction)



CNNs: POOLING

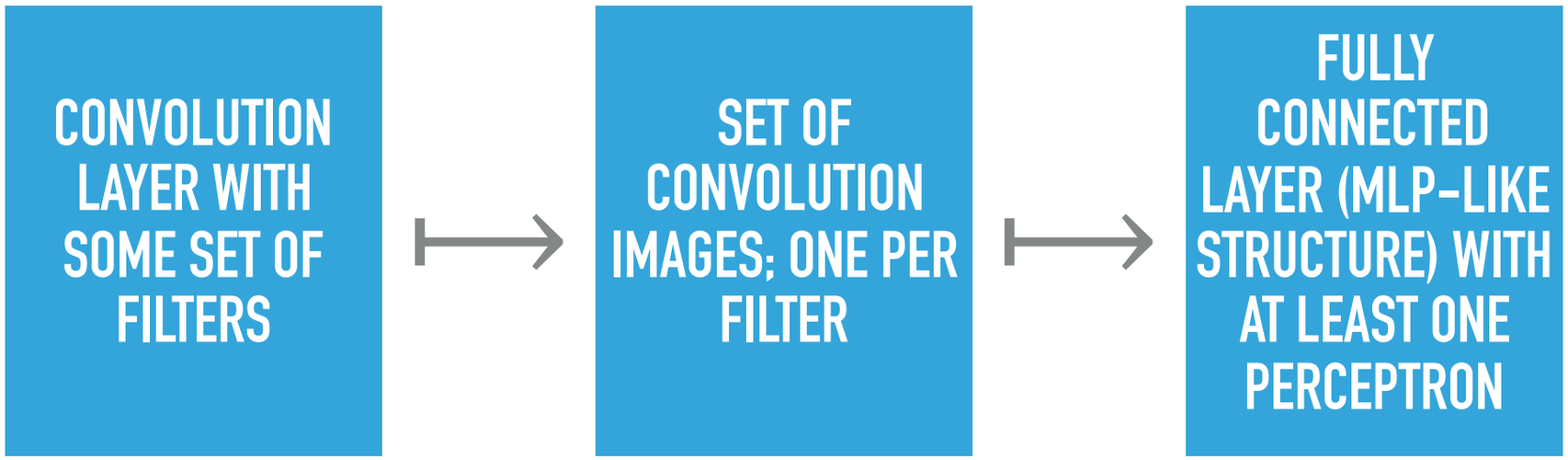
- The pooling process is applied to each individual part of the image (i.e. dimensional reduction)



and so on…

CNNs: ARCHITECTURES

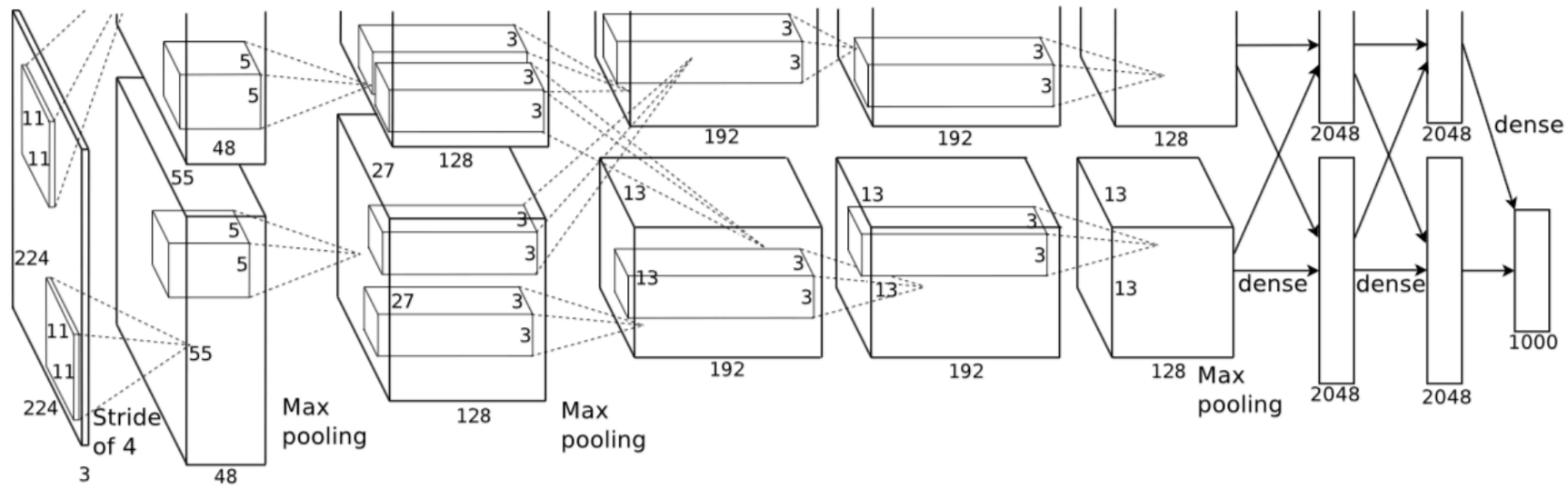
- The simplest convolutional neural network (CNN) architecture is:



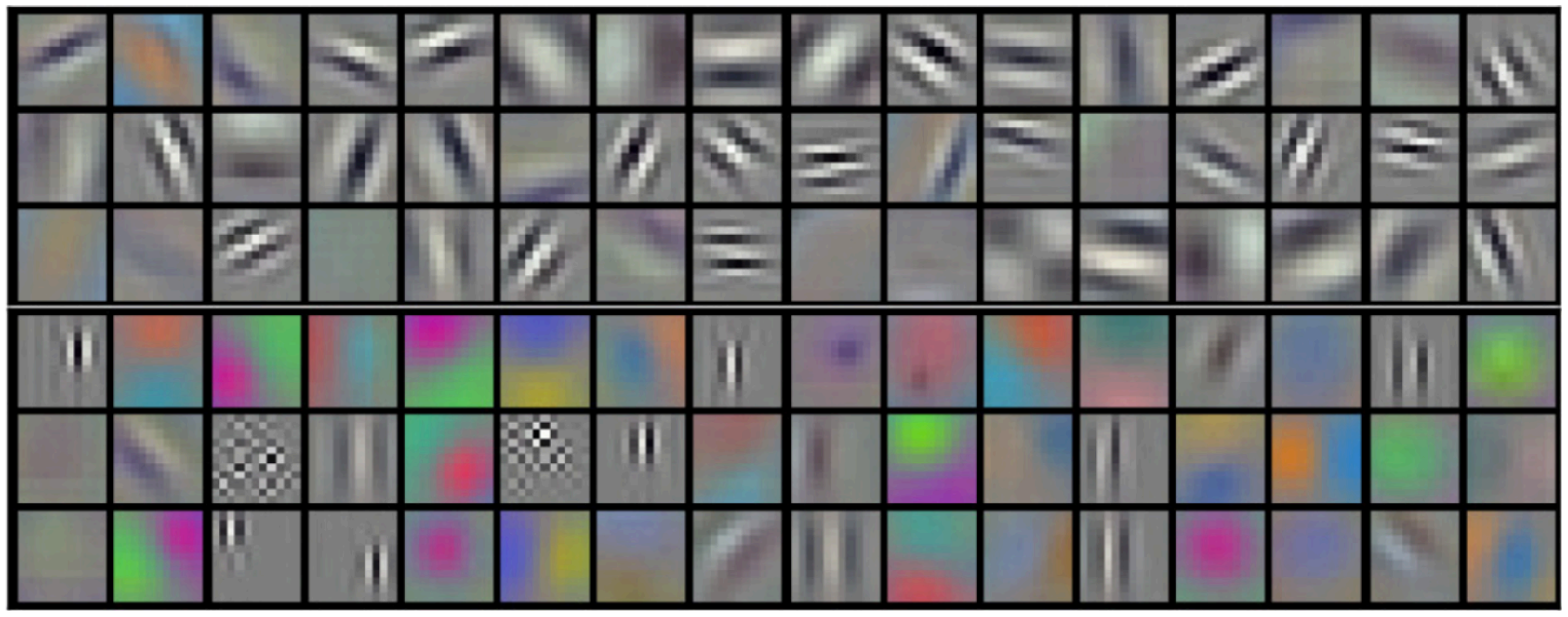
- The convolution layer takes an image and applies a set of k filters to the image.
- Each filter results in a new convolution image as its output.
- All of the features in all of the convolution images are combined to make a final combined output of the information.

CNNs: ARCHITECTURES

- e.g. the AlexNet CNN architecture



This is a complicated arrangement of convolution and pooling layers, with fully connected layers at the end (right hand side).

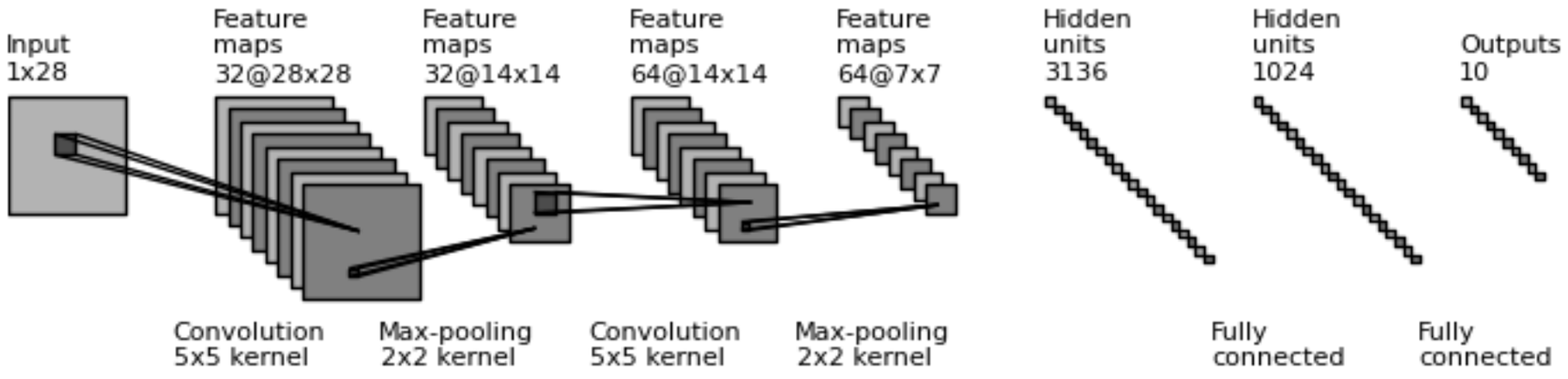


The convolution filters learned for this problem are able to pick out different forms and colours from an input example image.

Krizhevsky et al., Neural Information Processing Systems conference proceedings.

CNNs: ARCHITECTURES

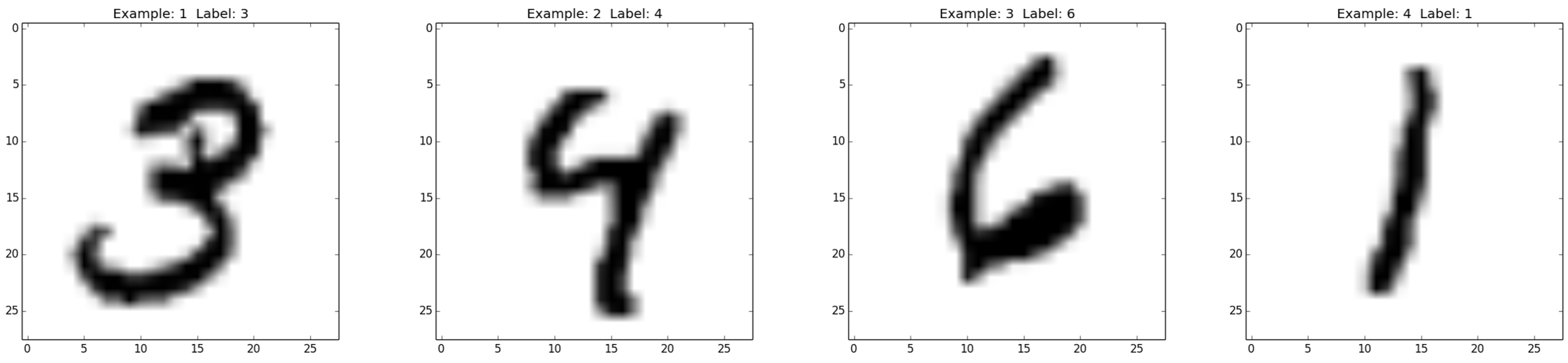
- We can include multiple convolution layers layers that may add to the information extracted from the image.
- We can add pooling layers to reduce the dimensionality.
- e.g. MNIST: handwritten numbers from 0 to 9.



- 28x28 input image (e.g. MNIST example).
- 2 convolution layers using 5x5 filter kernels.
- Each convolution layer followed by a 2x2 max-pooling layer.
- 2 fully connected layers leading to 10 outputs.

CNNs: ARCHITECTURES

- MNIST
  - Standard library of hand written numbers for benchmarking algorithms: 1, 2, 3, 4, 5, 6, 7, 8, 9, 0.
  - Images are 28x28 pixels (greyscale).
  - Several examples are shown below



[1] Neural Computation, Volume 22, Number 12, December 2010
http://yann.lecun.com/exdb/mnist/

CNNs: ARCHITECTURES/INPUT DATA

- So far we have focussed on monochrome images with a single number in the range [0, 1] representing each pixel.
- What about colour images?
  - These have 3 numbers (r, g, b) describing each pixel.
  - Trivial to extend the convolution and pooling processes to work on images of some arbitrary depth D (=3 for colour).
  - 3-fold increase in weight parameters to determine.
- e.g. CFAR10 benchmark training set[1].

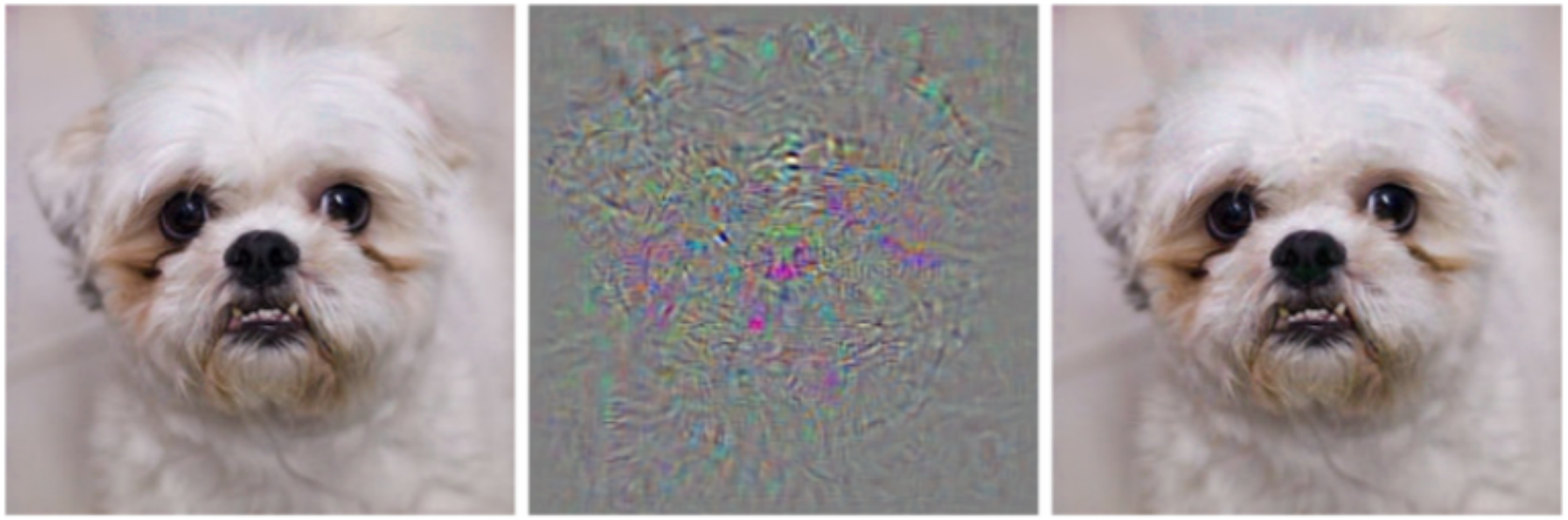
[1] https://www.cs.toronto.edu/~kriz/cifar.html

CNNs: ARCHITECTURES/INPUT DATA

- More abstract problems can be addressed in the same way.
- Some examples are given below:
  - Transient searches can be addressed by stacking images together to form an image of depth D.
  - Tracking problems can be addressed by stacking measurement data from successive layers.
  - More arbitrary problems can be addressed by feeding pixelised images of 2D correlations plots between pairs of input “features” can be constructed. Stacks of these can be fed into a CNN.

ADVERSARIAL NETWORKS

- Szegedy et al found that small perturbations in the example were sufficient to lead to example mis-classification.
  - The inclusion of some training example that has a small amount of noise added to it resulting in a change in classification is counter to the aim of obtaining a generalised performance from a network.



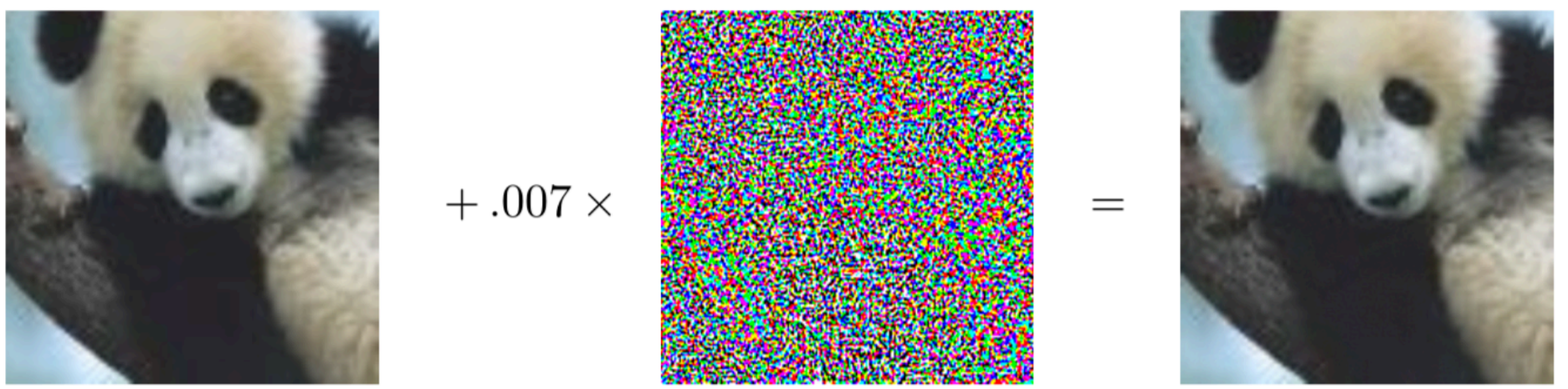
Correctly classified image | Perturbation of image | Incorrectly classified resultant image

From Szegedy et al,

Szegedy et al, ICLR, abs/1312.6199
Goodfellow et al, CoRR, abs/1412.6572, arXiv:1406.2661

ADVERSARIAL NETWORKS

- Szegedy et al found that small perturbations in the example were sufficient to lead to example mis-classification.
  - The inclusion of some training example that has a small amount of noise added to it resulting in a change in classification is counter to the aim of obtaining a generalised performance from a network.



From Goodfellow et al,

Szegedy et al, ICLR, abs/1312.6199
Goodfellow et al, CoRR, abs/1412.6572, arXiv:1406.2661

ADVERSARIAL NETWORKS

- Adversarial examples with small perturbations in the input are difficult for networks to classify because of their linear nature in high dimensional feature spaces.
  - e.g. for $w^T x \mapsto w^T (x + \eta)$ a large value of dim(x) will result in a large change in the contribution of the perturbed dot product.
- Adversarial training relies on a modification of the cost function with the intention that the use of adversarial examples in training regularise the optimisation process by identifying flaws in the model that is being learned.
- This in turn leads to an improved training performance.
  - Exploiting the nature of adversarial examples allowed Goodfellow et al., to reduce the error rate for image classification with MNIST data; beyond the benefits of using dropout.
  - The interpretation of this procedure is that one is “minimising the worst case error when the data are perturbed by an adversary”.

Szegedy et al, ICLR, abs/1312.6199
Goodfellow et al, CoRR, abs/1412.6572, arXiv:1406.2661

ADVERSARIAL NETWORKS

- The idea behind adversarial networks is to find some way to present adversarial examples alongside data to improve the ability of the model to recognise both the data and its adversarial counterpart.

- Train two models simultaneously:
  - **G: a generative model** (the model used to generate adversarial examples for training)
  - **D: a discriminative model** (the model used to make a prediction that an example is either data or from the generative model)
  - Train D to maximise the rate of correct outcomes for training examples and samples from the generative model.
  - Train G to minimise ln(1– D[G(z)])*.

- Over training epochs the generative model G will improve so that it mimics D better.

  * It can be problematic to train G in early epochs as it is possible for D to reject samples from G with high confidence; so for early epochs one can maximise ln(D[G(z)]) to overcome this limitation.

Szegedy et al, ICLR, abs/1312.6199
Goodfellow et al, CoRR, abs/1412.6572, arXiv:1406.2661 also see Goodfellow's NIPS proceedings on Generative Adversarial Networks: arxiv:1701.00160.

USING AUTO-ENCODERS

- Sometimes it is difficult to specify what features need to be extracted from input data to solve a particular problem, as the type of interest can manifest itself differently under different scenarios.
  - e.g. consider a particle representation in the laboratory. It's properties in the lab frame depend on the orientation;
    - if we know how to represent the data in terms of Lorentz invariants then we obtain a much clearer picture of the existence of underlying particles via their mass spectrum.
    - Using Lorentz invariants we are able to understand the spectrum of particles produced in collisions at a deeper level.
- In analogy Auto-encoders can learn representations of the data. They have two parts:
  - an encoder that maps input features into a different representation;
  - a decoder that is used to convert back into the original format.

USING AUTO-ENCODERS

- The purpose of the auto-encoder is to learn the mapping

$$x \quad \mapsto \quad h \quad \mapsto \quad r$$

$$h = f(x)$$

$$r = g(h) = g(f(x))$$

x: input feature space

h: hidden layer giving an alternate representation of the data

r: reconstruction of x computed by the auto-encoder

- If the auto-encoder learns to copy the input feature to the reconstructed output perfectly then r is not particularly useful.

- The representation given by h can be useful:
  - If dim(h) < dim(x) then the auto-encoder is under-compete and the auto-encoder learns how to copy x to r using the most important input features.
  - Under-complete auto-encoders can learn something interesting about the input data, which can be extracted from h.
  - auto-encoders with too large a dim(h) can learn to copy x without extracting any interesting information about the data: these are not interesting unless regularisation is used to change the behaviour.

USING AUTO-ENCODERS

- The purpose of the auto-encoder is to learn the mapping

$$x \quad \mapsto \quad h \quad \mapsto \quad r$$

$$h = f(x)$$

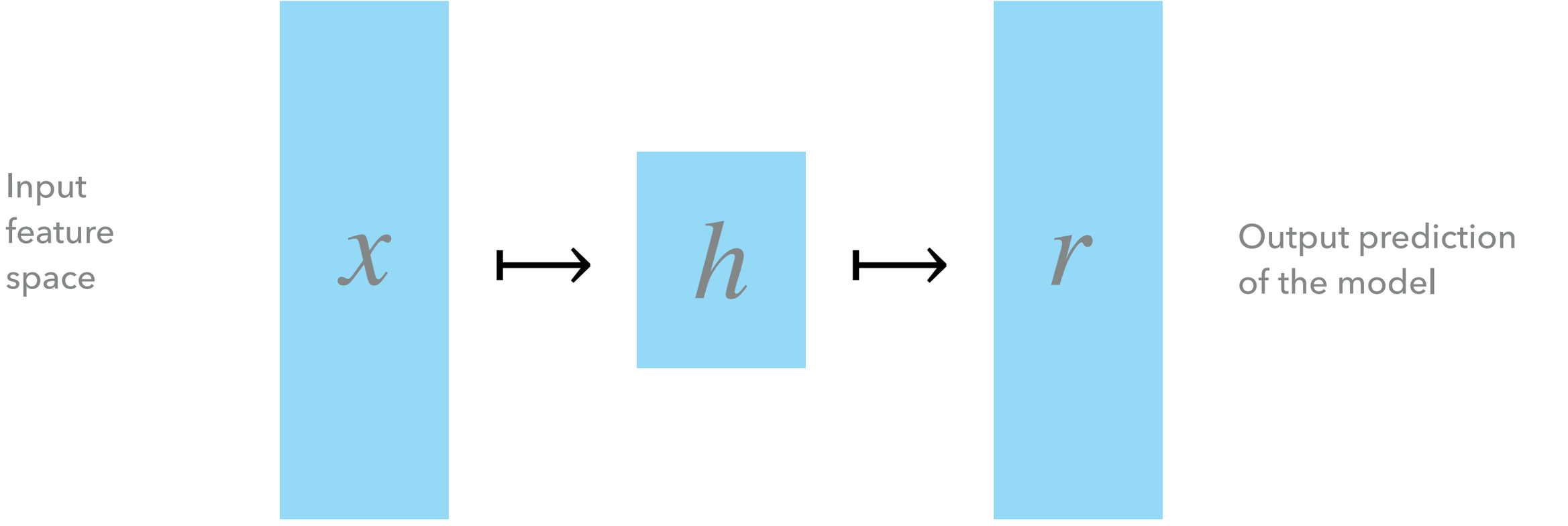
$$r = g(h) = g(f(x))$$

x: input feature space

h: hidden layer giving an alternate representation of the data

r: reconstruction of x computed by the auto-encoder

- If the auto-encoder learns to copy the input feature to the reconstructed output perfectly then r is not particularly useful.



USING AUTO-ENCODERS

- Denoising auto-encoders are extensions of auto-encoders, where the input x is replaced by $\tilde{x}$, which is a copy of the example x, modified according to some noise.
  - i.e. $$r = g(f(\tilde{x}))$$
- The loss function used to train an auto-encoder compares x with r; e.g. using an MSE or L2-norm loss function this is simply a sum of squared difference.
- This type of auto encoder allows a network to learn how to reconstruct x taking into account the noise (e.g. a systematic uncertainty).

EXAMPLES: JET FLAVOUR CLASSIFICATION

- Quark colour confinement leads to the creation of jets as pairs of quarks and anti-quarks are pulled apart.
  - As a result we don't see bare quarks.
  - However we do see many hadrons eliminating from some underlying quark.
- These hadrons form objects called jets that are reconstructed in our detectors.
- The nature of the underlying quark is of interest as knowing that allows us to infer something about an underlying interaction; e.g. the decay of a Higgs boson to two b-quarks requires that we accurately identify events with two (or more) b jets.

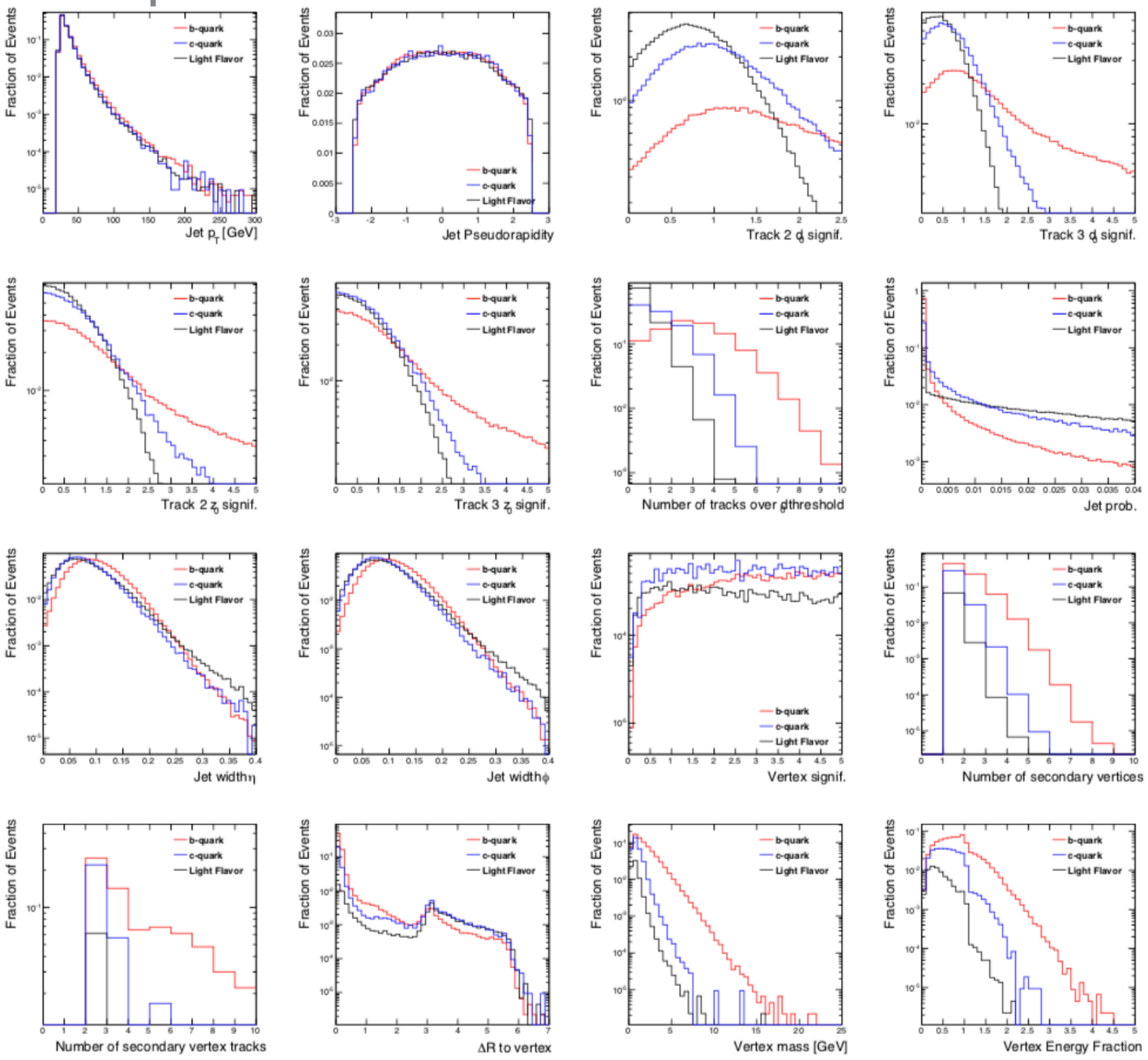
EXAMPLES: JET FLAVOUR CLASSIFICATION

- Consider jet identification in pp collisions at the LHC.
- Vertex, track and calorimeter information are used to identify jets.
- Aim: separate jets into:
  - light quarks (u, d, s);
  - charm;
  - beauty.
- Guest's study uses the anti-kt algorithm for jet reconstruction and FastJet.
- 8 million jets for training, 1 million for testing and 1 million for validation.

Guest et al., Phys. Rev. D **94**, 112002 (2016)

EXAMPLES: JET FLAVOUR CLASSIFICATION

- Input variables:



- The d_0 and z_0 significance of the 2nd and 3rd tracks attached to a vertex, ordered by d_0 significance.
- The number of tracks with d_0 significance greater than 1.8σ.
- The JetProb [32] light jet probability, calculated as the product over all tracks in the jet of the probability for a given track to have come from a light-quark jet.
- The width of the jet in η and ϕ, calculated for η as

$$\left(\frac{\sum_i p_{\mathrm{T}i}\Delta\eta_i^2}{\sum_i p_{\mathrm{T}}}\right)^{1/2}$$

  and analogously for ϕ.
- The combined vertex significance,

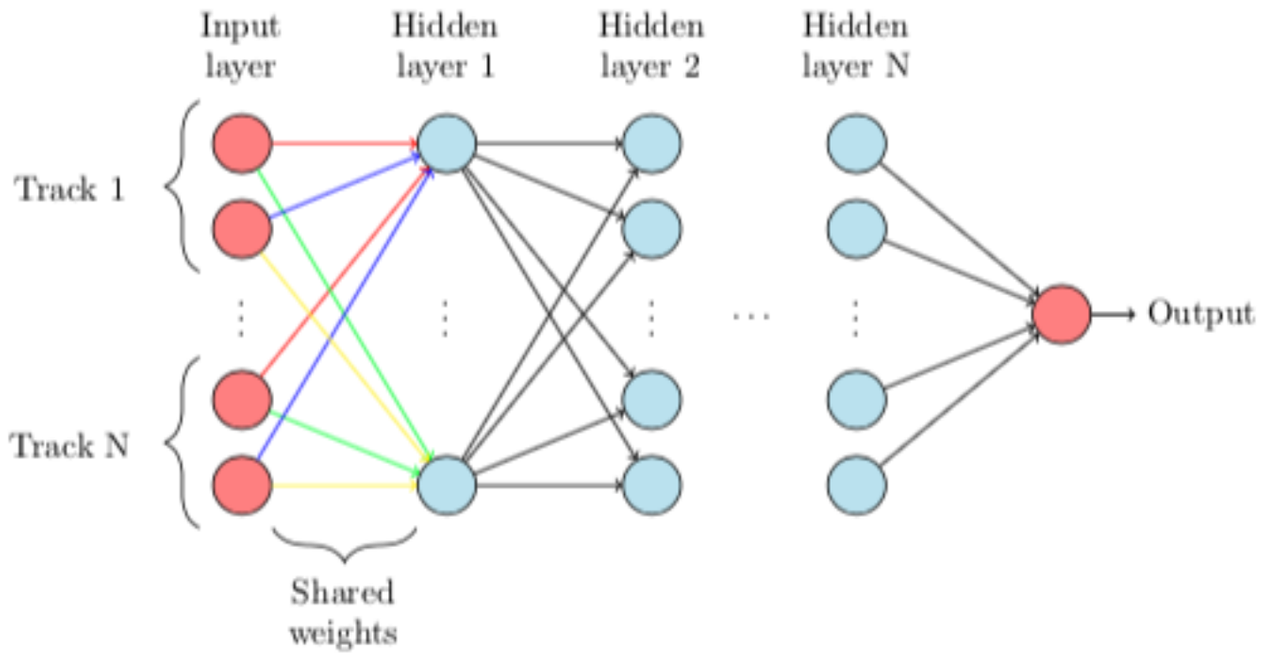
$$\frac{\sum_i d_i/\sigma_i^2}{\sqrt{\sum_i 1/\sigma_i^2}}$$

  where d is the vertex displacement and σ is the uncertainty in vertex position along the displacement axis.
- The number of secondary vertices.
- The number of secondary-vertex tracks.
- The angular distance ΔR between the jet and vertex.
- The decay chain mass, calculated as the sum of the invariant masses of all reconstructed vertices, where particles are assigned the pion mass.
- The fraction of the total track energy in the jet associated to secondary vertices [1]

Guest et al., Phys. Rev. D **94**, 112002 (2016)

EXAMPLES: JET FLAVOUR CLASSIFICATION

- Several different algorithms are used including MLPs



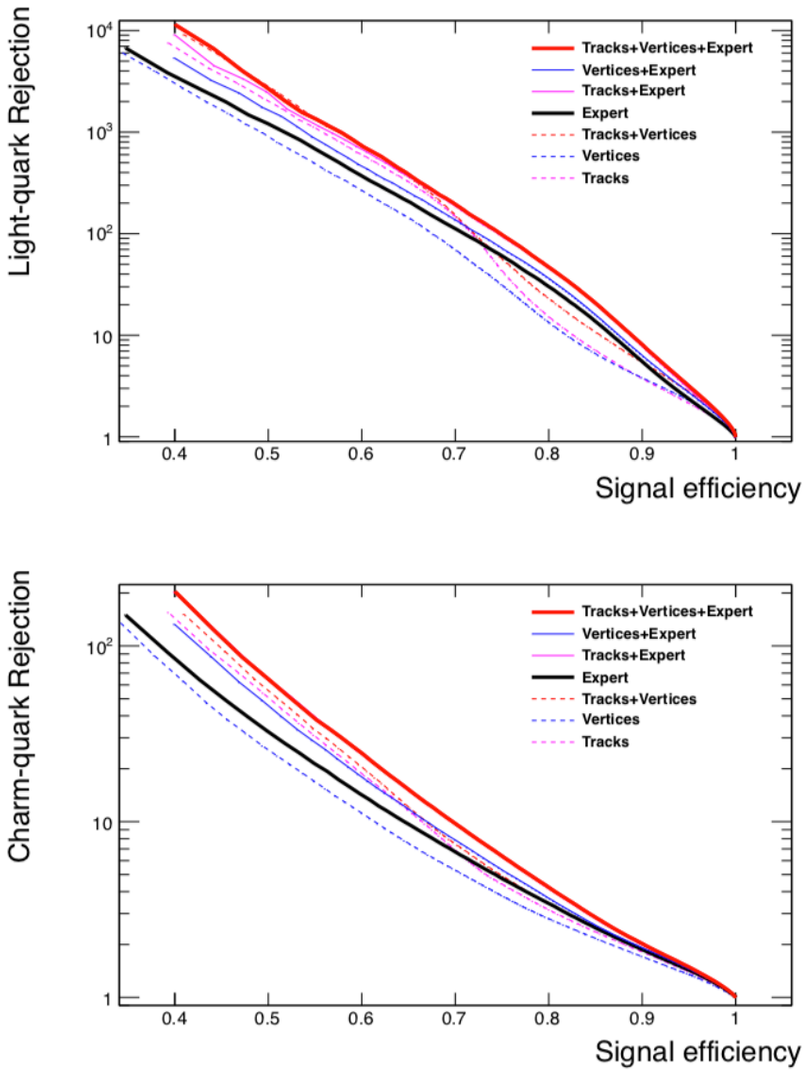
- "Experts" are networks that are trained to address a specific issue. This study constructs "Experts" that are used as inputs to a final network.

Guest et al., Phys. Rev. D **94**, 112002 (2016)

EXAMPLES: JET FLAVOUR CLASSIFICATION

- Several different algorithms are used including MLPs

Inputs			Technique	AUC
Tracks	Vertices	Expert		
✓			Feedforward	0.916
✓			LSTM	0.917
✓			Outer	0.915
	✓		Feedforward	0.912
	✓		LSTM	0.911
	✓		Outer	0.911
✓	✓		Feedforward	0.929
✓	✓		LSTM	0.929
✓	✓		Outer	0.928
		✓	Feedforward	0.924
		✓	LSTM	0.925
		✓	Outer	0.924
✓		✓	Feedforward	0.937
✓		✓	LSTM	0.937
✓		✓	Outer	0.936
	✓	✓	Feedforward	0.931
	✓	✓	LSTM	0.930
	✓	✓	Outer	0.929
✓	✓	✓	Feedforward	0.939
✓	✓	✓	LSTM	0.939
✓	✓	✓	Outer	0.937

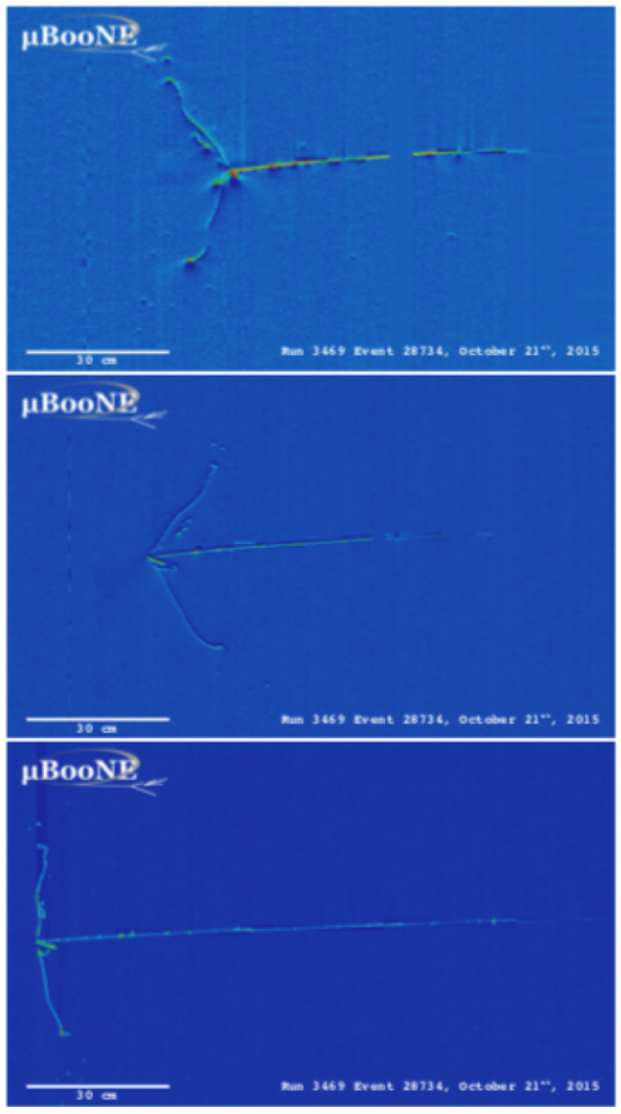


- They give similar performance.
- CMS has followed a deep network approach to jet tagging in their latest work (e.g. H→bb) [see A.M. Sirunyan et al 2018 JINST 13 P05011, CMS PAS HIG-18-016].

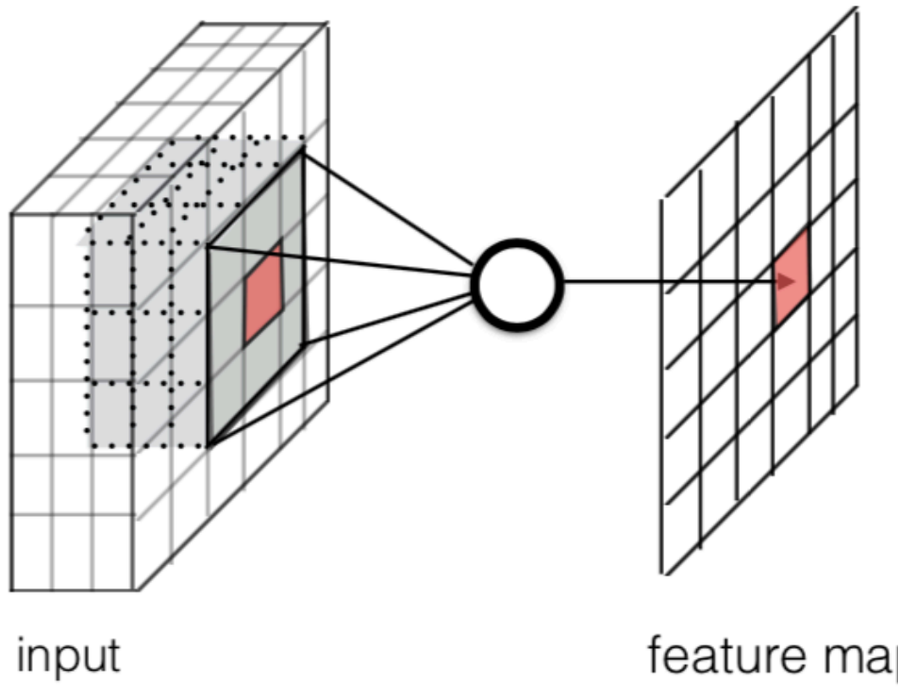
Guest et al., Phys. Rev. D **94**, 112002 (2016)

EXAMPLES: PARTICLE IDENTIFICATION AT MICROBOONE

- MicroBooNE LAr TPC produces images that need to be interpreted in terms of the underlying neutrino interaction physics:



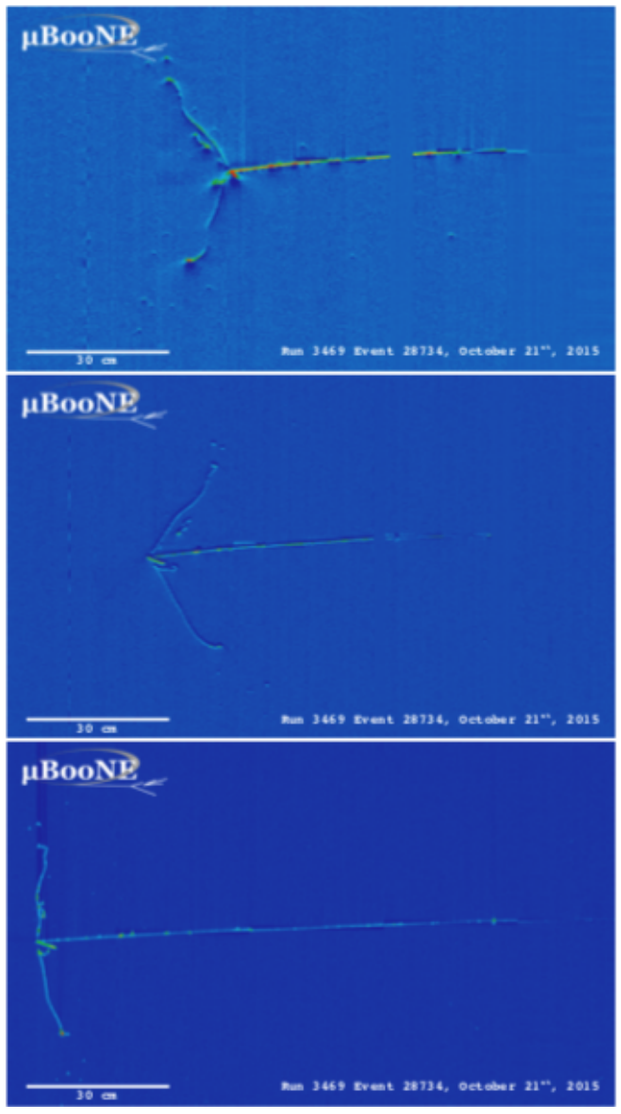
  - 3 different views of the same ν interaction.
  - Use existing software available from the web with many of the techniques discussed in these lectures.



MicroBooNE Collaboration, JINST **12** (2017) no.03, P03011

EXAMPLES: PARTICLE IDENTIFICATION AT MICROBOONE

- MicroBooNE LAr TPC produces images that need to be interpreted in terms of the underlying neutrino interaction physics:



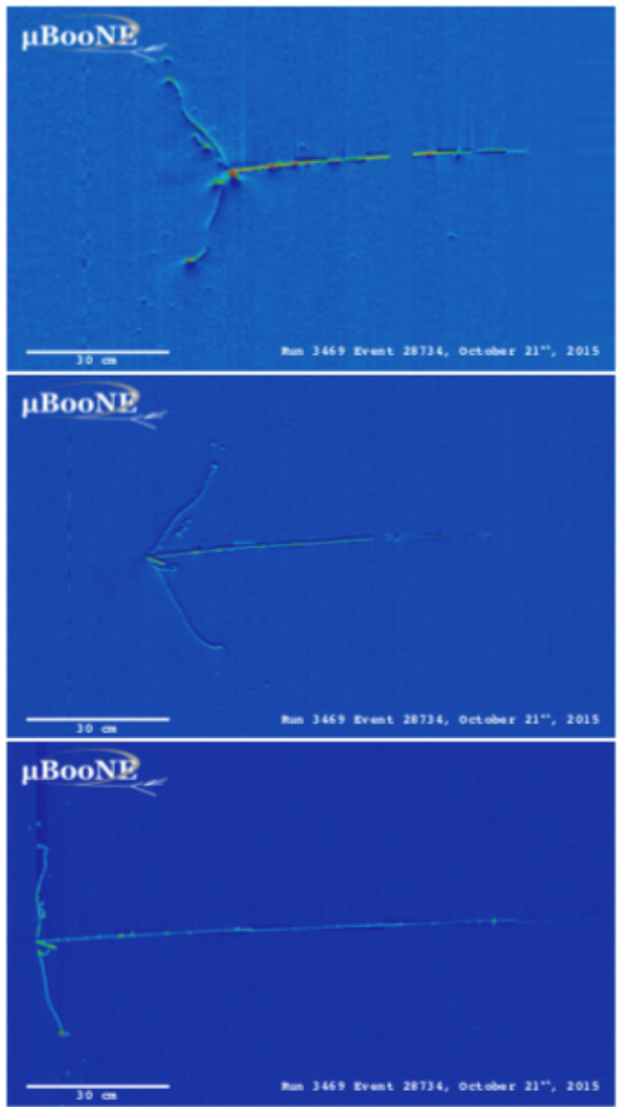
  - 3 different views of the same ν interaction.
  - Use existing software available from the web with many of the techniques discussed in these lectures.

Software	ref.	Purpose	Used in Demonstrations
LArSoft	[7]	Simulation and Reconstruction	1-3
uboonecode	[8]	Simulation and Reconstruction	1-3
LArCV	[9]	Image Processing and Analysis	1-3
Caffe	[10]	CNN Training and Analysis	1-3
AlexNet	[1]	Network Model	1,2
GoogLeNet	[11]	Network Model	1
Faster-RCNN	[12]	Network Model	1,2
Inception-ResNet-v2	[13]	Network Model	2
ResNet	[14]	Network Model	3

MicroBooNE Collaboration, JINST **12** (2017) no.03, P03011

EXAMPLES: PARTICLE IDENTIFICATION AT MICROBOONE

- MicroBooNE LAr TPC produces images that need to be interpreted in terms of the underlying neutrino interaction physics:



- Image resolution matters for the performance of this convolutional neural network.

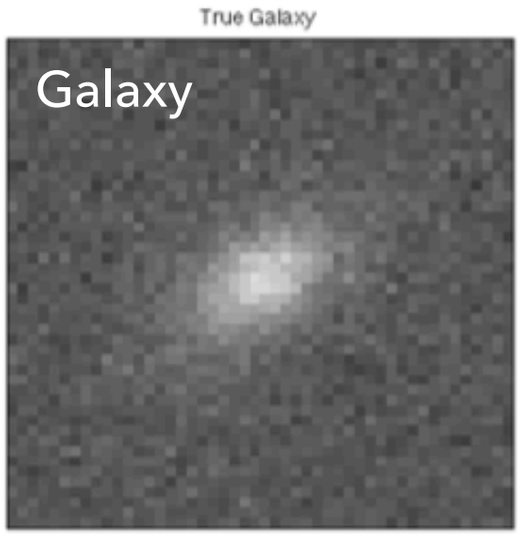
	Classified Particle Type				
Image, Network	e^- [%]	γ [%]	μ^- [%]	π^- [%]	proton [%]
HiRes, AlexNet	73.6 ± 0.7	81.3 ± 0.6	84.8 ± 0.6	73.1 ± 0.7	87.2 ± 0.5
LoRes, AlexNet	64.1 ± 0.8	77.3 ± 0.7	75.2 ± 0.7	74.2 ± 0.7	85.8 ± 0.6
HiRes, GoogLeNet	77.8 ± 0.7	83.4 ± 0.6	89.7 ± 0.5	71.0 ± 0.7	91.2 ± 0.5
LoRes, GoogLeNet	74.0 ± 0.7	74.0 ± 0.7	84.1 ± 0.6	75.2 ± 0.7	84.6 ± 0.6

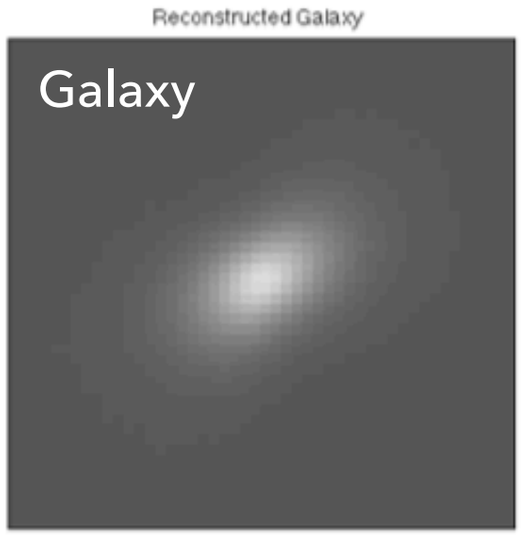
Table 2. Five particle classification performances. The very left column describes the image type and network where *HiRes* refers to a standard 576 by 576 pixel image while *LowRes* refers to a downsized image of 288 by 288 pixels. The five remaining columns denote the classification performance per particle type. Quoted uncertainties are purely statistical and assume a binomial distribution.

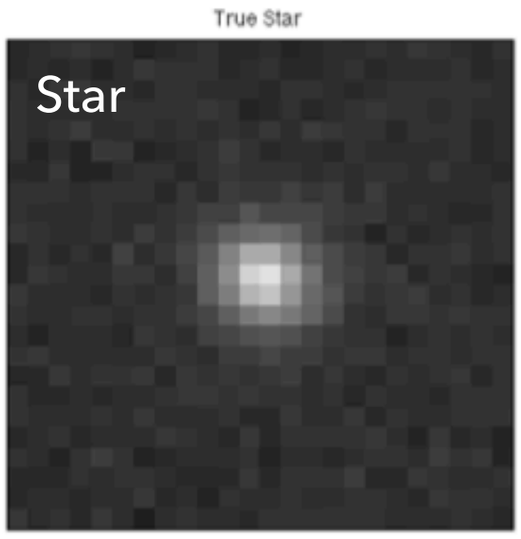
MicroBooNE Collaboration, JINST **12** (2017) no.03, P03011

EXAMPLES: AUTO-ENCODER

- Dimensional reduction using an auto-encoder configuration of 10 nodes in a hidden layer:
  - Galaxies can be described by 4 parameters (two ellipticities, a position angle and an amplitude).
  - Starts can be described by 2 parameters (radius and amplitude)
  - This auto-encoder configuration is able to reconstruct an image of the galaxy and star with noise removed.







Graff et al., Mon.Not.Roy.Astron.Soc. **441** (2014) no.2, 1741-1759

SUMMARY

- Deep learning is not a panacea; but it does have a role in machine learning.
  - Generally requires larger training/test data sets than other algorithms in order to avoid overtraining.
  - Can outperform other algorithms on some problems.
- Deep learning algorithms are more complicated than older algorithms; that makes it harder to optimise the HPs for them.
  - Considered deep MLPs and CNNs
  - Adversarial examples and networks have been discussed as a method for improving performance.
  - The role of auto-encoders in deep networks has also been introduced.

SUGGESTED READING (HEP RELATED)

- A lot of people are starting to use deep learning in high energy physics. The following is a selection of recent papers that you might wish to read at:
  - Guest et al., Phys. Rev. D 94, 112002 (2016) [jet classification]
  - A.M. Sirunyan et al 2018 JINST 13 P05011 [jet classification at CMS]
  - MicroBooNE Collaboration, JINST 12 (2017) no.03, P03011[LAr TPC]
  - Andrews et al., arXiv:1807.11916 [Event classification at CMS]
  - Guillen et al., arXiv:1807.09024 [air shower analysis]
  - Baldi et al., Nature Commun. 5 (2014) 4308 [Searches]

SUGGESTED READING (NON-HEP)

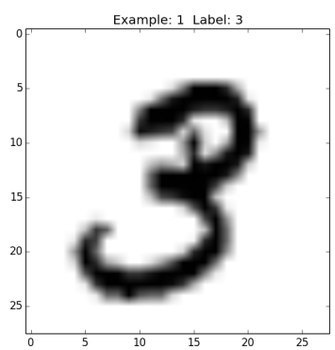
- Please see the references mentioned in the slides. There are a number of good books on deep learning that are available; and two examples are given here:
  - Goodfellow et al, Deep Learning, MIT Press (2016) [from the perspective of methods and algorithms]
  - A. Géron, Hands on Machine Learning with Scikit-Learn and TensorFlow, O'Reilly (2017) [from the perspective of coding up and using these methods]

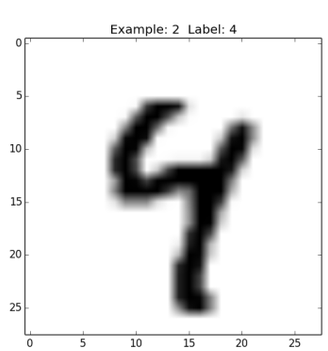
APPENDIX

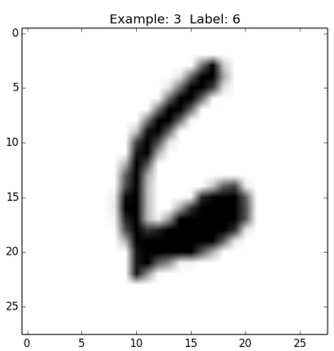
- The remainder of these slides discuss in brief the MNIST and CFAR-10(0) samples.

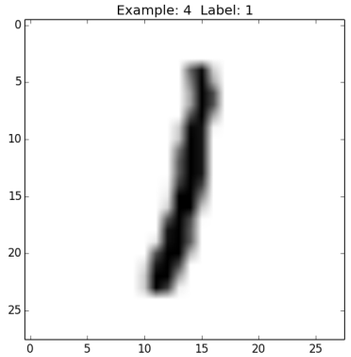
MNIST

- For more than two classification output types we need to have N_{type} perceptrons in the final output layer.
  - Each output perceptron has an all or nothing response that classifies if a training example is classified as that type or not.
  - e.g. the numbers 1, 2, 3, … 9, 0 [MNIST example]









- If we have a complete set of possible outcomes then we can use this constraint to reduce the number of perceptrons to N_{type}.
  - Assumes that the default classification for one category is given by an example not being classified as any of the others.

http://yann.lecun.com/exdb/mnist/

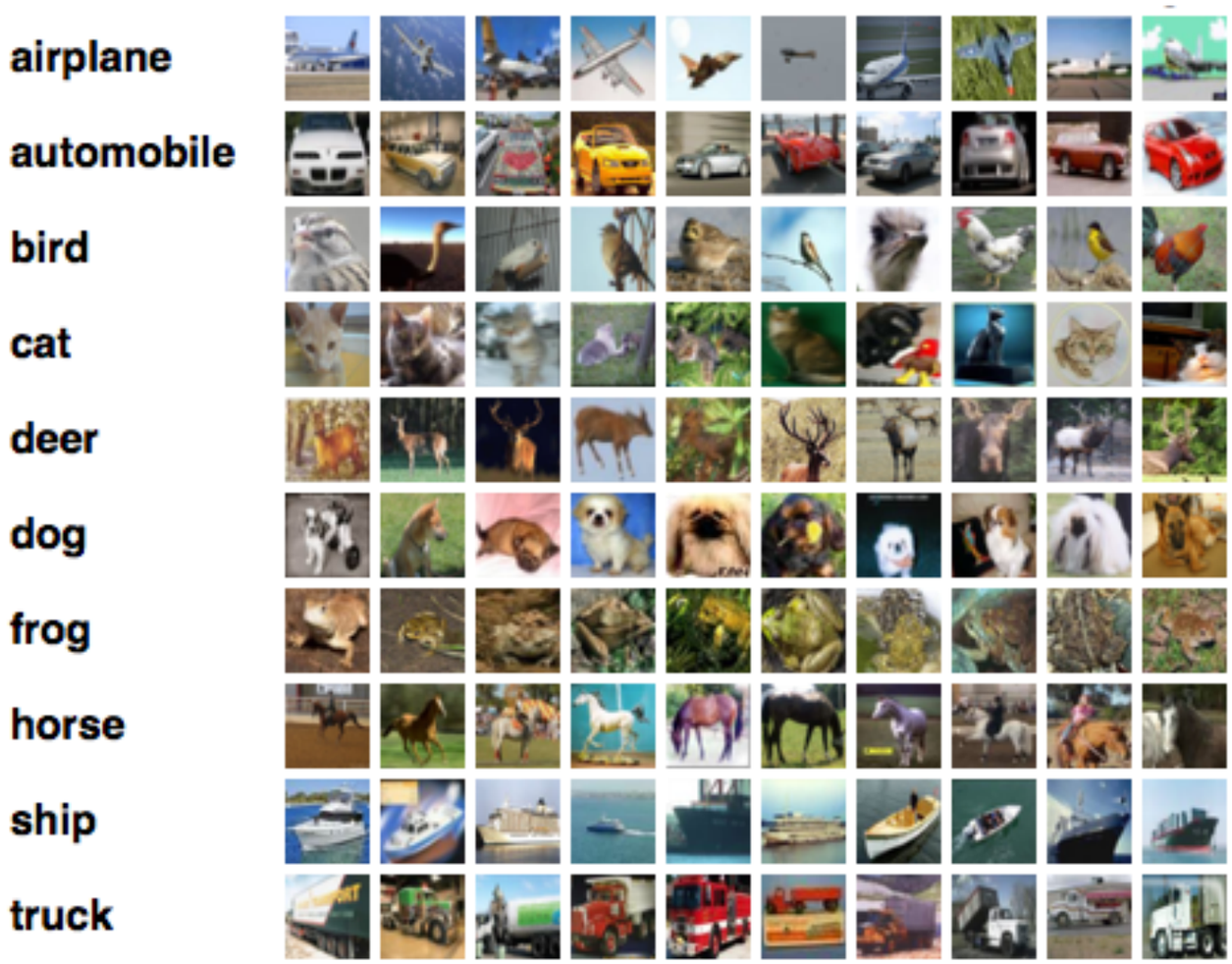
MNIST

- 60000 training examples
- 10000 test examples
- These are greyscale images (one number required to represent each pixel)
  - Renormalise [0, 255] on to [0, 1] for processing.
- Each image corresponds to a 28x28 pixel array of data.
  - For an MLP this translates to 784 features.

http://yann.lecun.com/exdb/mnist/

CFAR-10

- 60k 32x32 colour images (so each image is a tensor of dimension 32x32x3).
  - This is a labelled subset of an 80 million image dataset.
- 10 classes:



https://www.cs.toronto.edu/~kriz/cifar.html

CFAR-100

- 100 class variant on the CFAR10 sample:
  - 32x32 colour images (so each image is a tensor of dimension 32x32x3).
- 100 classes:

Superclass	Classes
aquatic mammals	beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food containers	bottles, bowls, cans, cups, plates
fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
household electrical devices	clock, computer keyboard, lamp, telephone, television
household furniture	bed, chair, couch, table, wardrobe
insects	bee, beetle, butterfly, caterpillar, cockroach
large carnivores	bear, leopard, lion, tiger, wolf
large man-made outdoor things	bridge, castle, house, road, skyscraper
large natural outdoor scenes	cloud, forest, mountain, plain, sea
large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
medium-sized mammals	fox, porcupine, possum, raccoon, skunk
non-insect invertebrates	crab, lobster, snail, spider, worm
people	baby, boy, girl, man, woman
reptiles	crocodile, dinosaur, lizard, snake, turtle
small mammals	hamster, mouse, rabbit, shrew, squirrel
trees	maple, oak, palm, pine, willow
vehicles 1	bicycle, bus, motorcycle, pickup truck, train
vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

https://www.cs.toronto.edu/~kriz/cifar.html

CFAR-100

- From this course you have the technical ability to work with these data sets;
  - CNNs
  - MLPs
- The issue you have to solve is reading them in; there are examples of how to do this on the tensor flow website:
  - https://www.tensorflow.org/tutorials/images/deep_cnn

https://www.cs.toronto.edu/~kriz/cifar.html