PRACTICAL MACHINE LEARNING

CLASSIFICATION

Queen Mary
University of London

THIS COURSE – 4 WORKSHOPS (LECTURE+EXERCISES)

- Lecture 1 (Monday) Introduction to machine learning with neural networks and linear regression
- Lecture 2 (Tuesday) Optimisation and non-linear regression with neural networks
- Lecture 3 (Wednesday) Classification and convolutional neural networks for image classification
- Lecture 4 (Thursday) Robustness and adversarial examples to image classification problems

WHO WE ARE

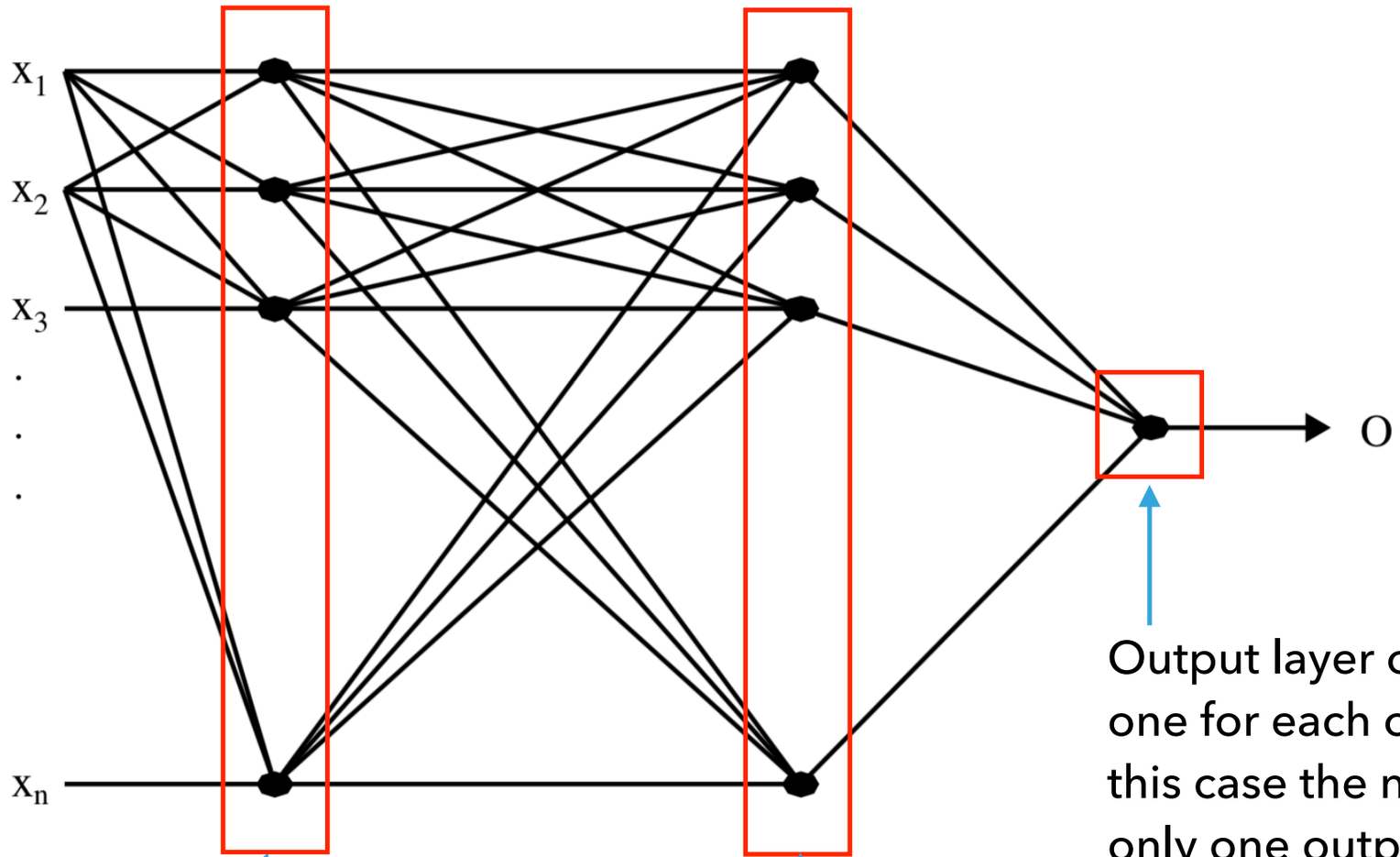
Dr Alex Booth [he/him]

Dr Linda Cremonesi [she/her]

Dr Abbey Waldron [she/her]

CLASSIFICATION

- From the introductory NNs Lecture



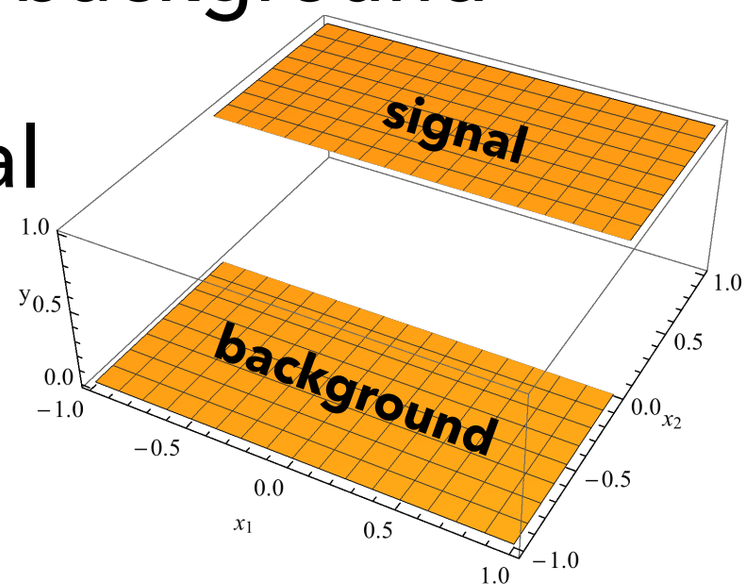
Input layer of n perceptrons; one for each dimension of the input feature space

Hidden layer of some number of perceptrons, M; at least one for each dimension of the input feature space.

Output layer of perceptrons; one for each output type. In this case the network has only one output.

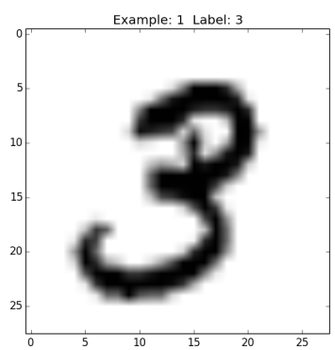
CLASSIFICATION

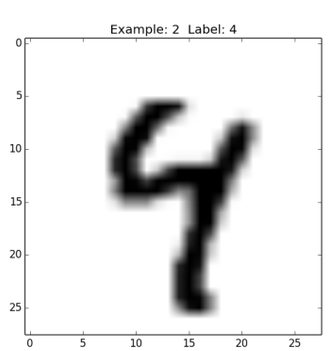
- The final perceptron in a network can be used to assign a type to a data example.
  - e.g. consider a binary activation function that gives an all or nothing response with a data sample that contains two types of event (signal and background)
    - Let nothing correspond to one type: background
    - Let all correspond to the other: signal
- Multi-class output is discussed shortly.

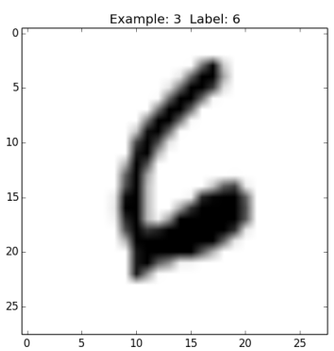


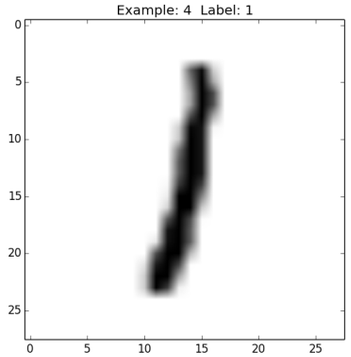
EXAMPLES: MNIST

- 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/

EXAMPLES: MNIST

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

* Depends on which activation function is being used.
http://yann.lecun.com/exdb/mnist/

EXAMPLES: MNIST

- The N_{type} = 10 perceptrons are used to make the following decisions:
  - The number 1 vs not the number 1
  - The number 2 vs not the number 2
  - The number 3 vs not the number 3
  - The number 4 vs not the number 4
  - The number 5 vs not the number 5
  - The number 6 vs not the number 6
  - The number 7 vs not the number 7
  - The number 8 vs not the number 8
  - The number 9 vs not the number 9
  - The number 0 vs not the number 0

For those with a statistical background, this is like a null hypothesis and an alternative hypothesis.

The null hypothesis provides a specific response/expectation.

The alternative hypothesis is the complement of the null.

In this context you classify an example as a specific type, or you provide a decision that it is not that type.

We will see more of the MNIST data when talking about convolutional neural networks.

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

EXAMPLES: MNIST

- An alternative representation is to use a softmax activation function to encode the 10 outputs in a single function.

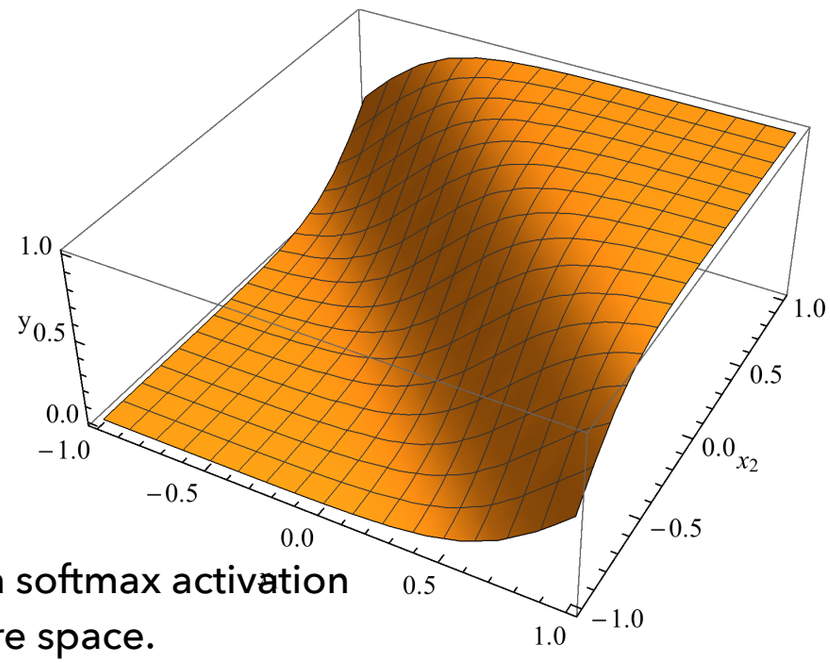
$$f_j(x) = \frac{e^{w_j^T x}}{\sum_{i=1}^{N} e^{w_i^T x}}$$

i is the index for the output classification type

The score for the i^{th} output is normalised by the sum of outputs.

$$f(x) = \frac{1}{\sum_{i=1}^{N} e^{w_i^T x}} \begin{bmatrix} e^{w_1^T x} \\ e^{w_2^T x} \\ \vdots \\ e^{w_N^T x} \end{bmatrix}$$

$f_i(x)$ is normalised to lie in the range [0, 1]



Example of the i^{th} output of a softmax activation function for a 2D input feature space.

- Can convert output to {0, 1}.

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

EXAMPLES: MNIST

- LeCun's website lists a number of complicated ways to train a neural network to solve this problem.

- Recent advances in computing have allowed the use of GPU's have meant that MLPs have been applied to the MNIST data, and have produced good results: error rate of 0.35% (Ciresan et al [1]).

ID	architecture (number of neurons in each layer)	test error for best validation [%]	best test error [%]	simulation time [h]	weights [milions]
1	1000, 500, 10	**0.49**	0.44	23.4	1.34
2	1500, 1000, 500, 10	**0.46**	0.40	44.2	3.26
3	2000, 1500, 1000, 500, 10	**0.41**	0.39	66.7	6.69
4	2500, 2000, 1500, 1000, 500, 10	**0.35**	0.32	114.5	12.11
5	9×1000, 10	**0.44**	0.43	107.7	8.86

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

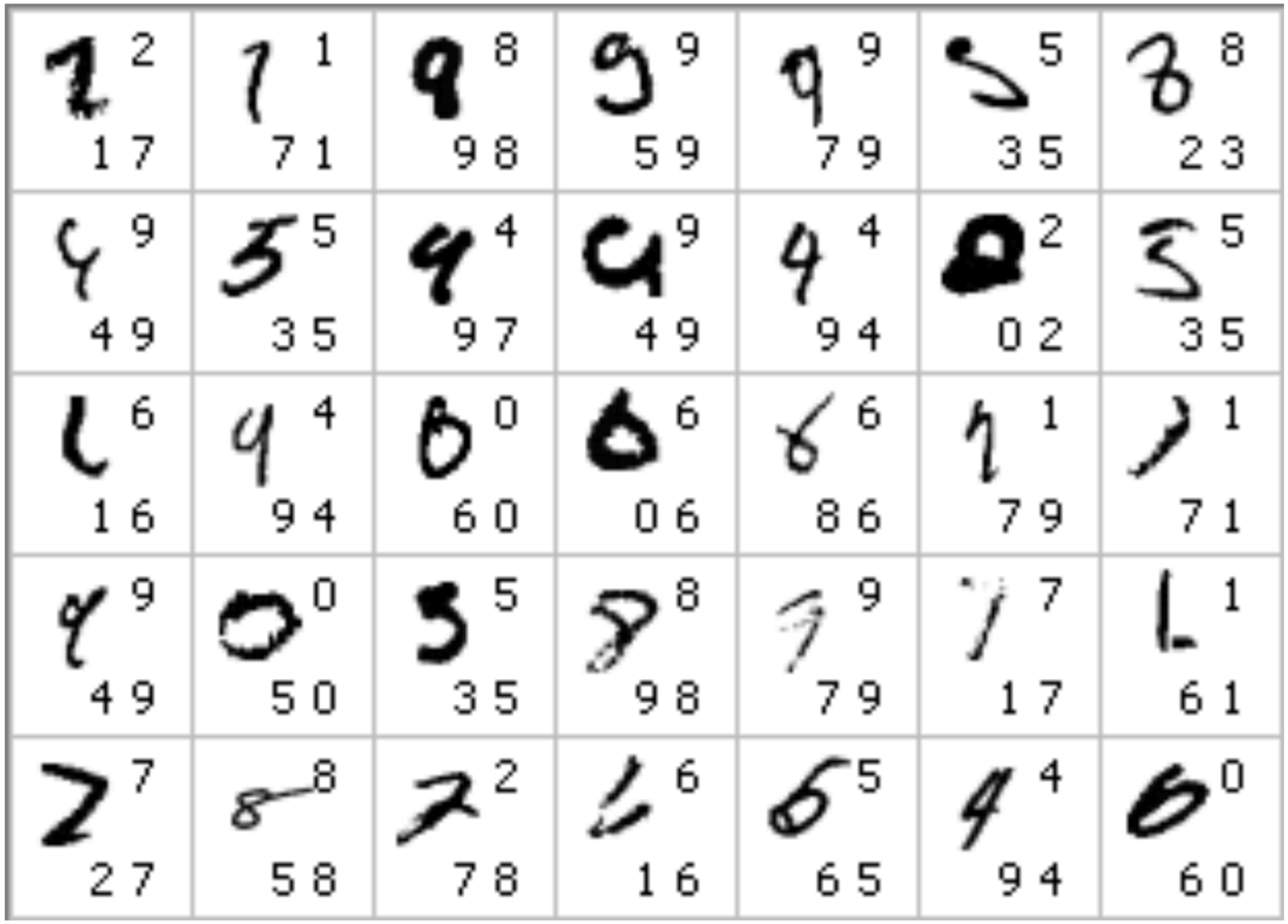
EXAMPLES: MNIST

- 35 training examples were mis-classified by the best NN architecture

Correct label

NUMBER

Two most likely predictions (L, R)



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

Queen Mary University of London

PRACTICAL MACHINE LEARNING

CNNS

LECTURE PLAN

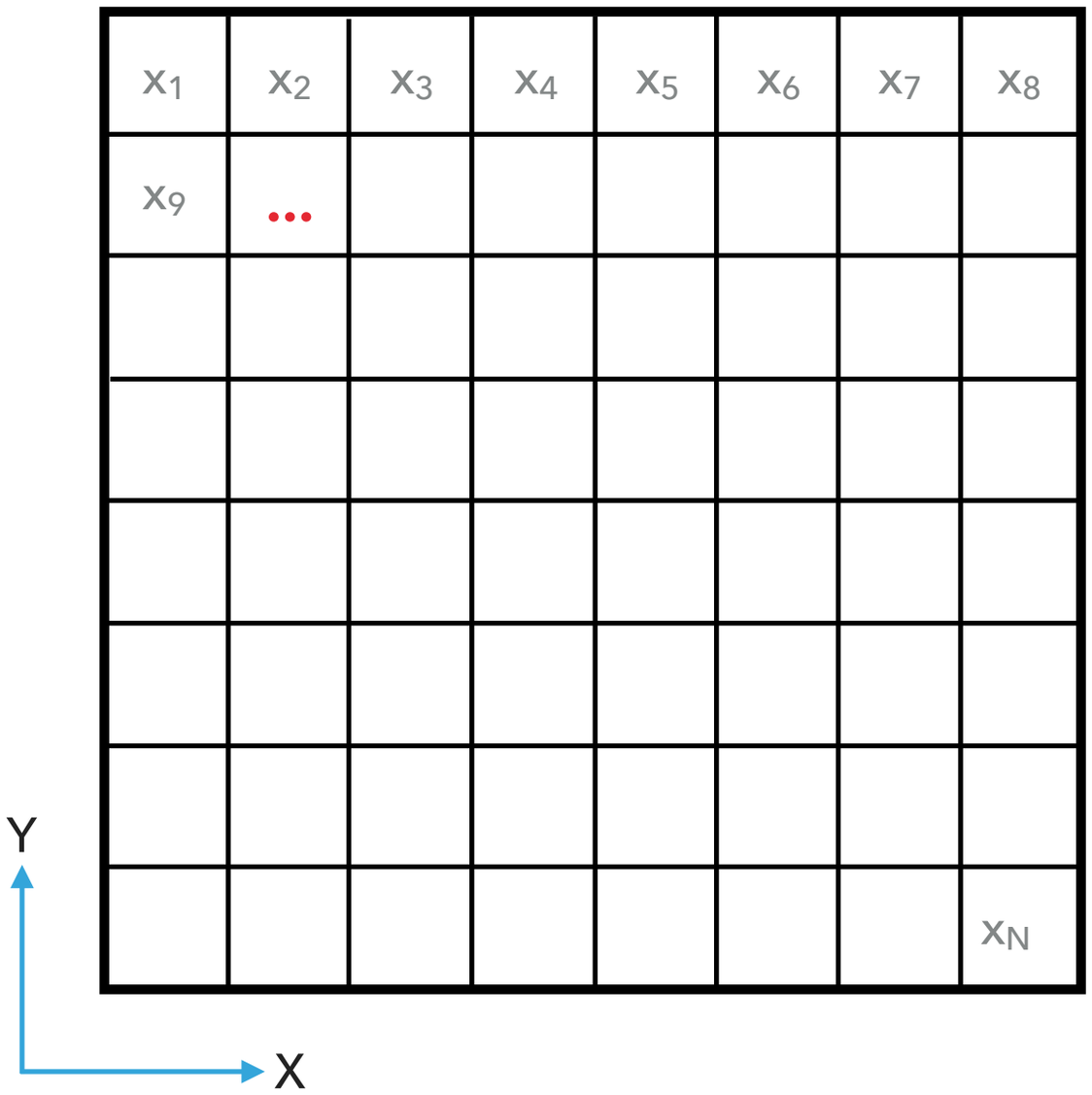
- Input data
- Convolution layers
  - Padding
- Pooling
  - Max-pooling and average pooling
- Convolutional Neural Network (CNN) architectures
  - Revisit input data
- Dropout
- Summary

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.

INPUT DATA

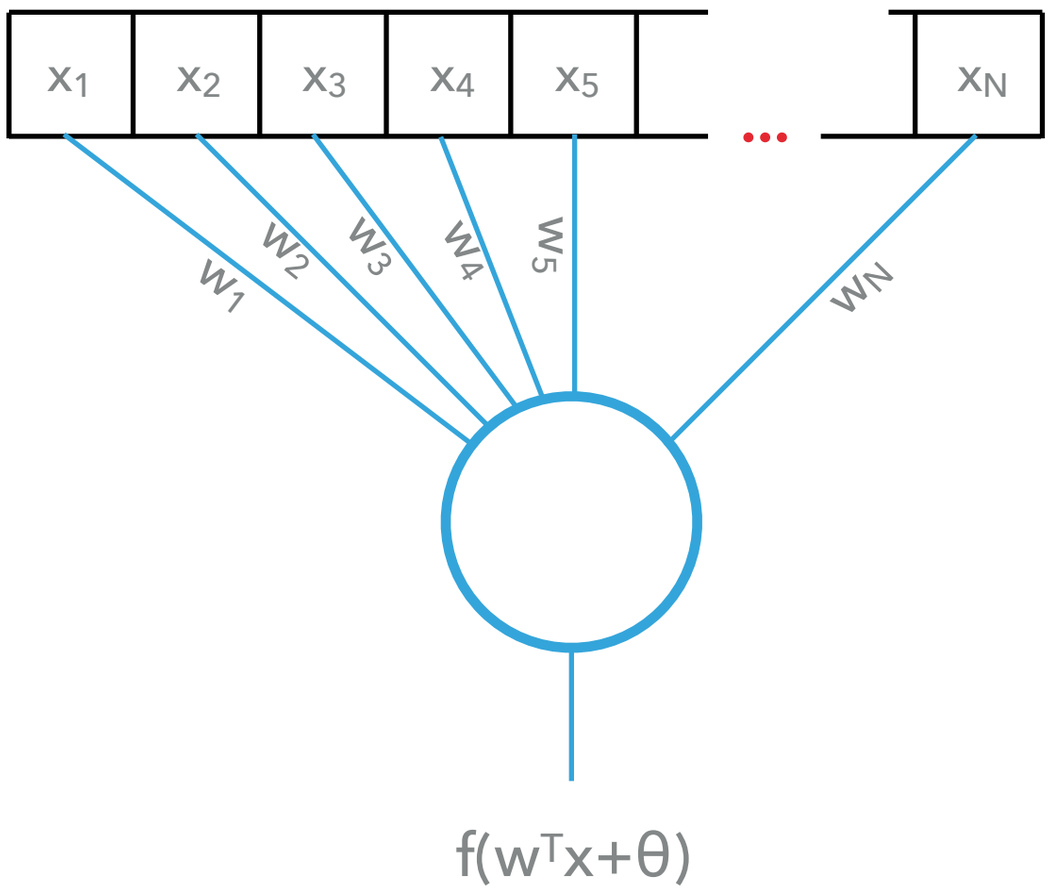


This is an 8 by 8 array of pixels, that corresponds to a 64 dimensional feature space.

- 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. $f(w^T x+\theta)$.

[1]Typically CNNs are applied to square images.

INPUT DATA

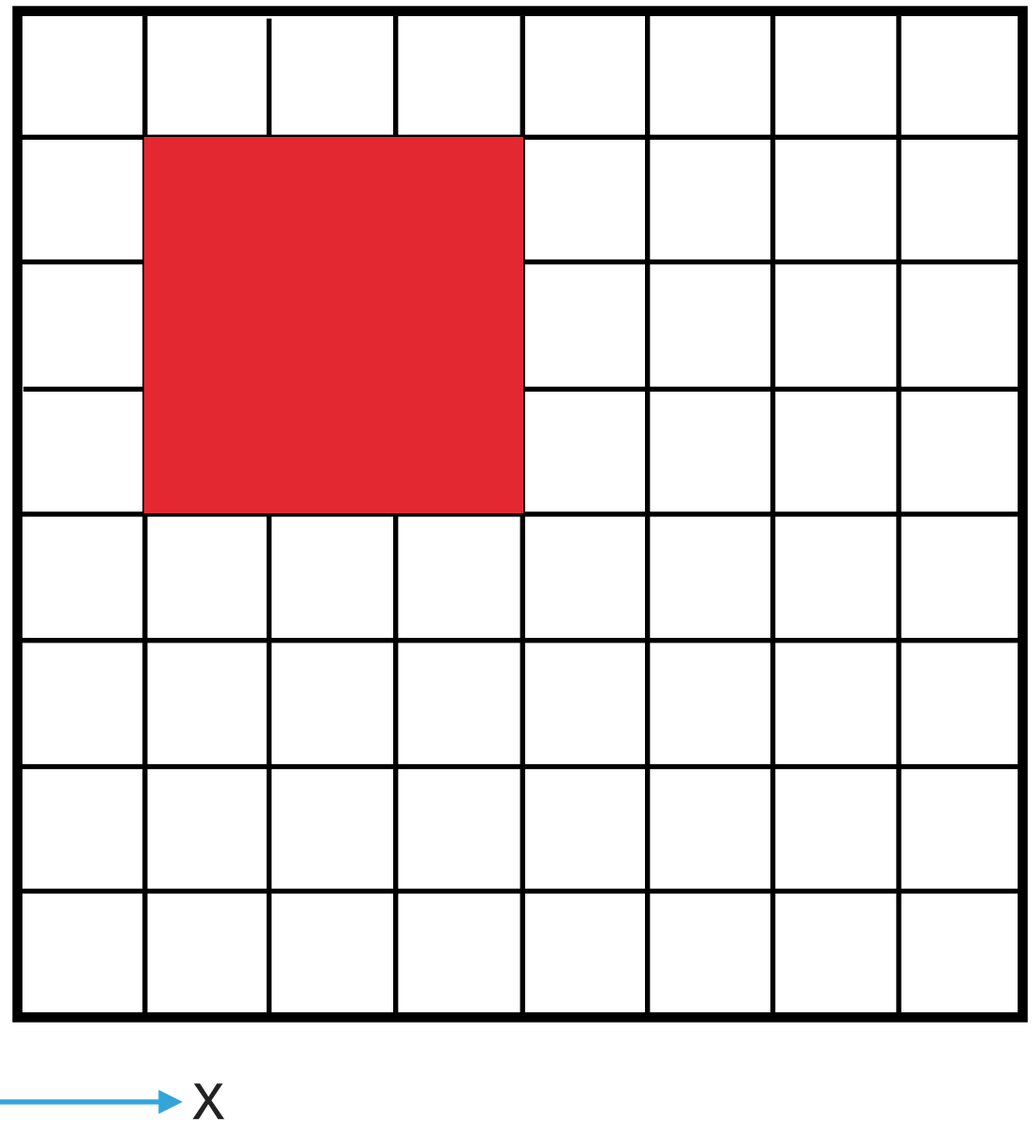


This is an 8 by 8 array of pixels, that corresponds to a 64 dimensional feature space.

- 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. $f(w^T x+\theta)$.

[1]Typically CNNs are applied to square images.

CONVOLUTION LAYERS

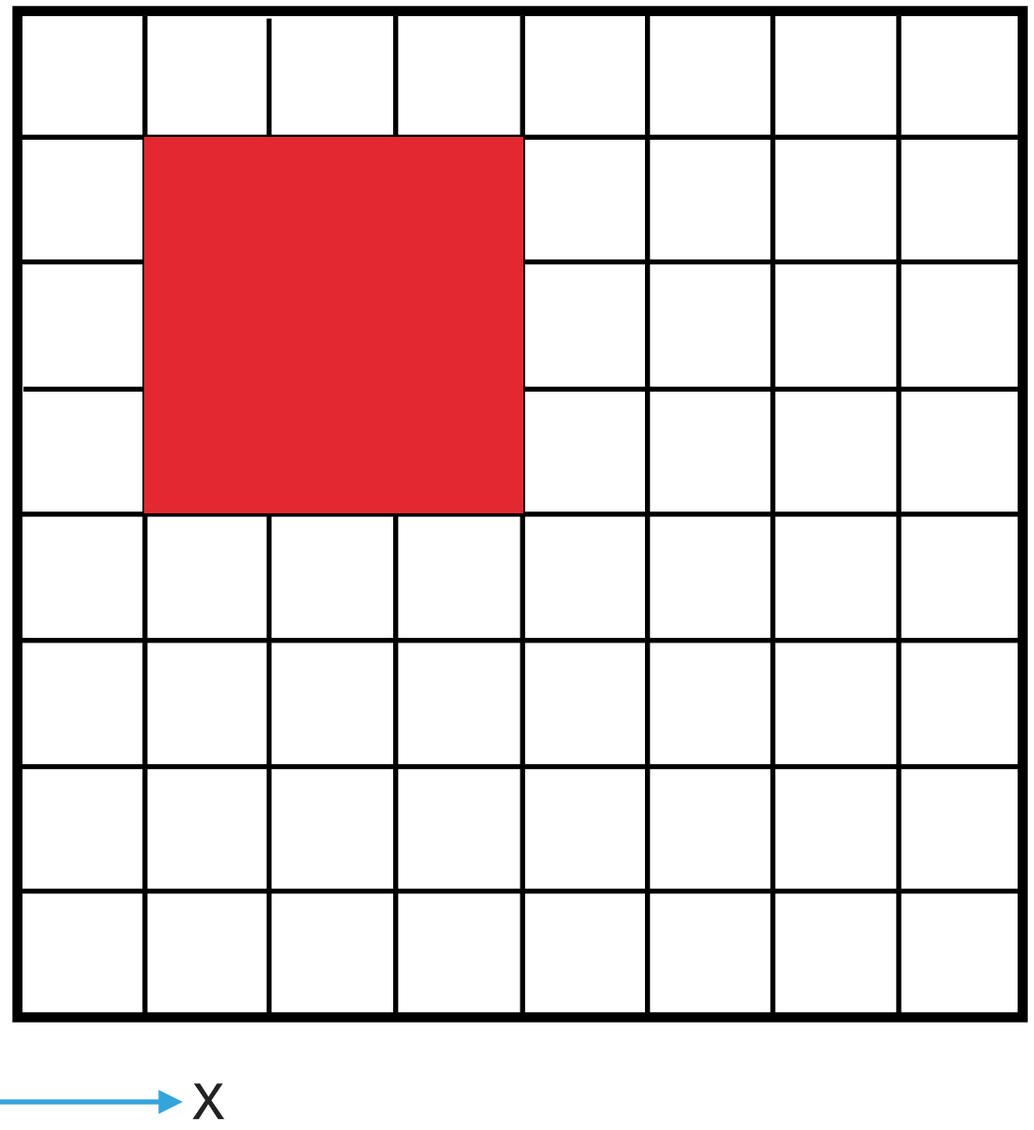


This is an 8 by 8 array of pixels, that corresponds to a 64 dimensional feature space.

- 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.

[1]Typically CNNs are applied to square images.

CONVOLUTION LAYERS



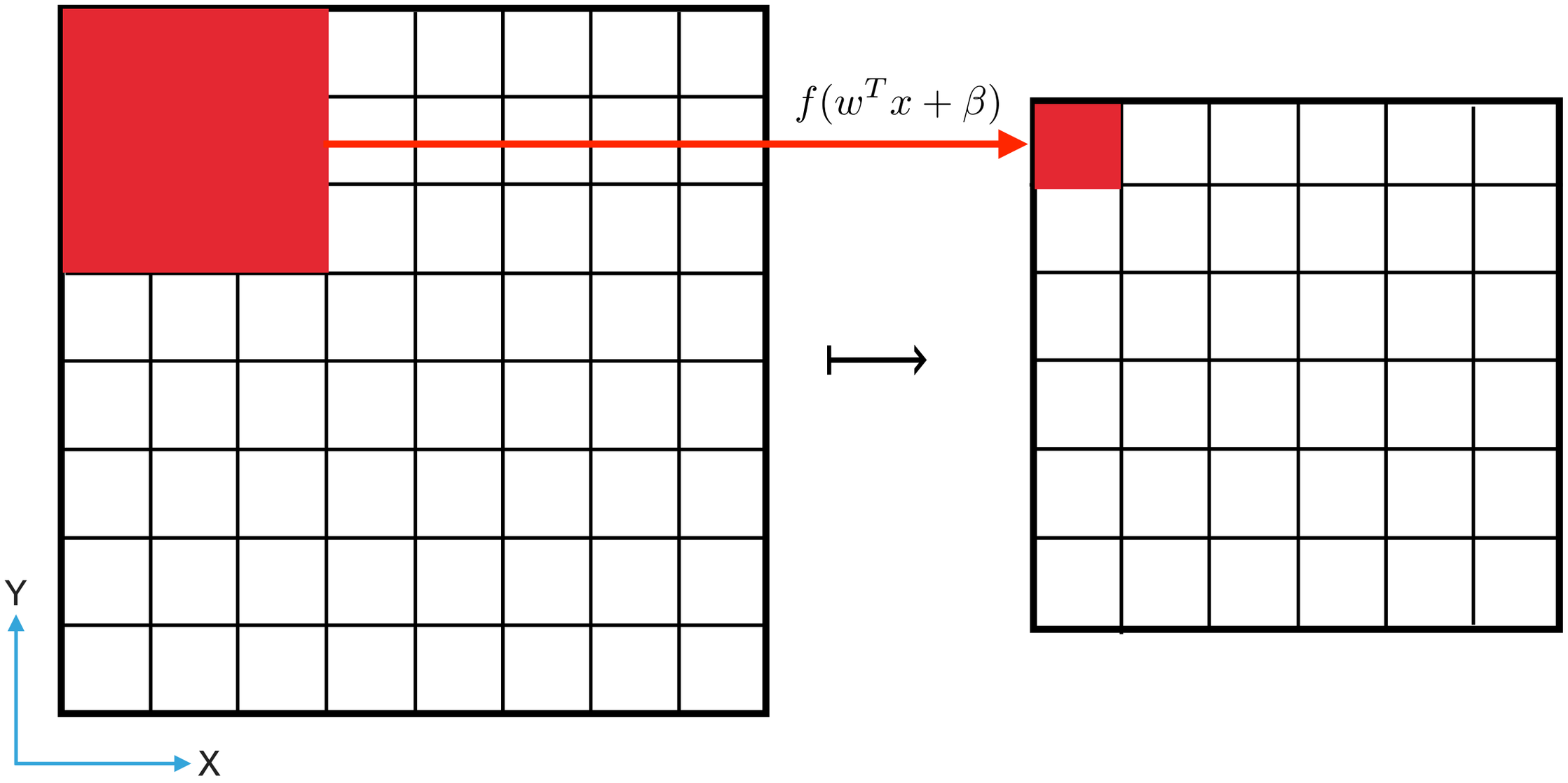
This is an 8 by 8 array of pixels, that corresponds to a 64 dimensional feature space.

- 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".

[1]Typically CNNs are applied to square images.

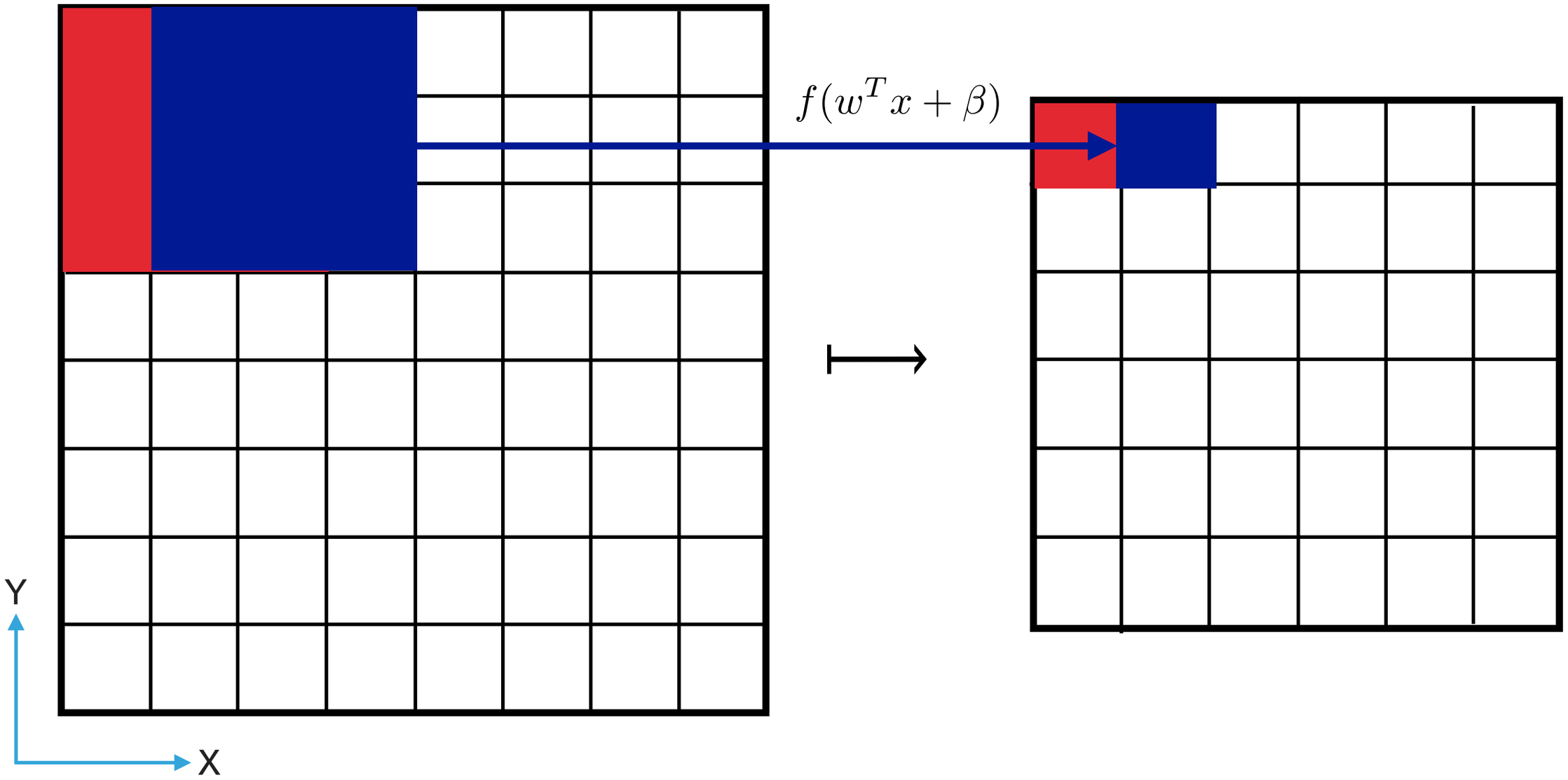
CONVOLUTION LAYERS

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



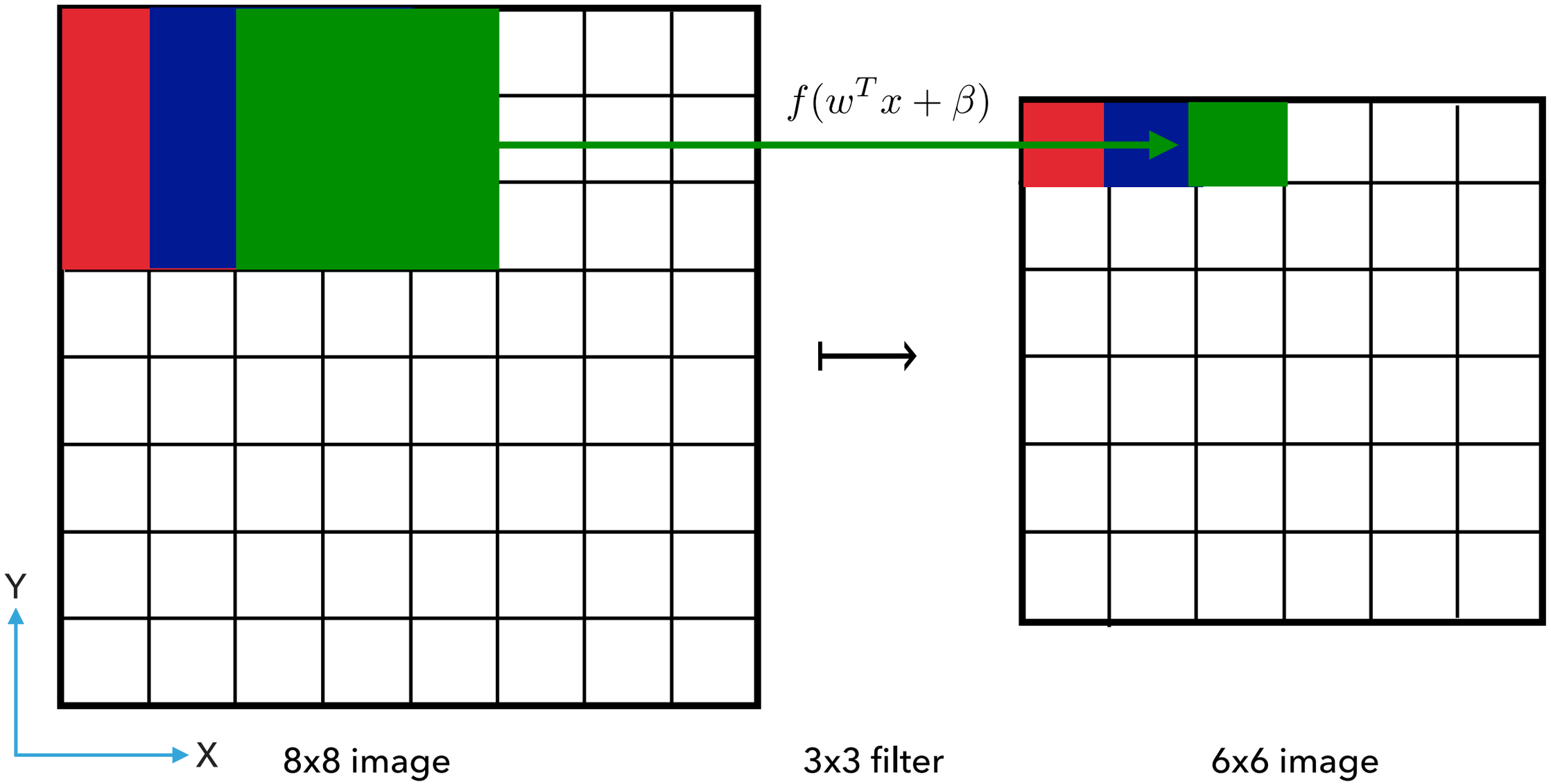
CONVOLUTION LAYERS

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



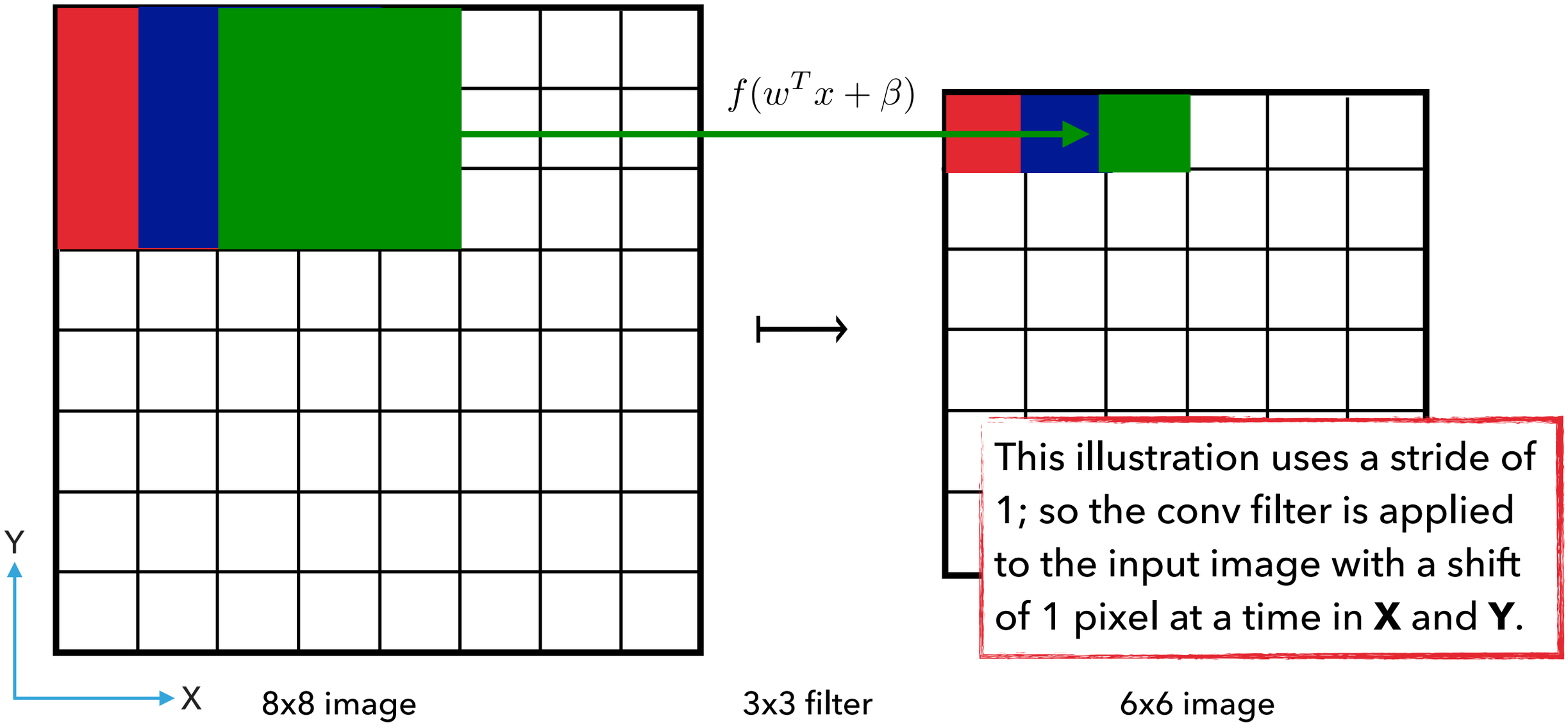
CONVOLUTION LAYERS

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



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**.

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.

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 convolved with the filter any number of times (within resource limitations) ***without reducing the dimensionality of the input feature space that contains non-trivial information***.

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.

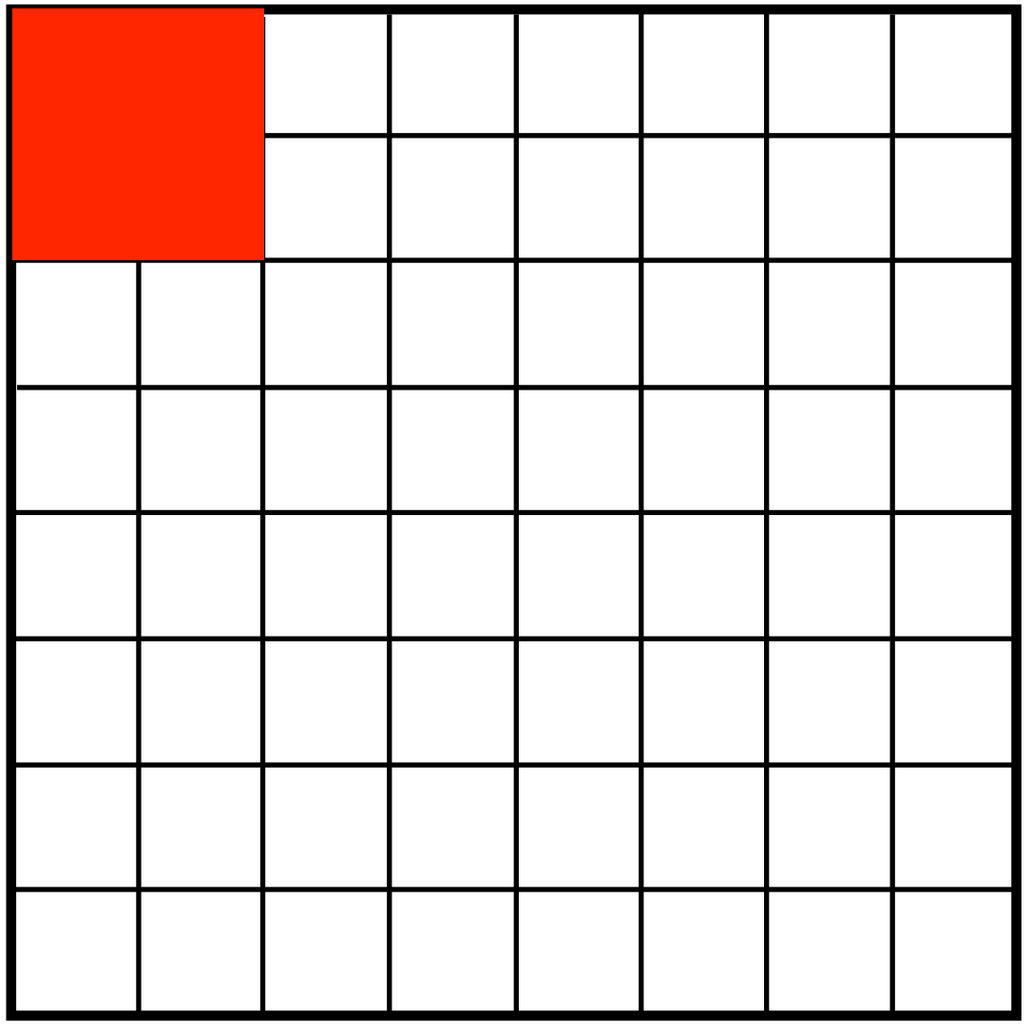
POOLING

- 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)

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.

$$f = \frac{1}{N} \sum_i x_i$$

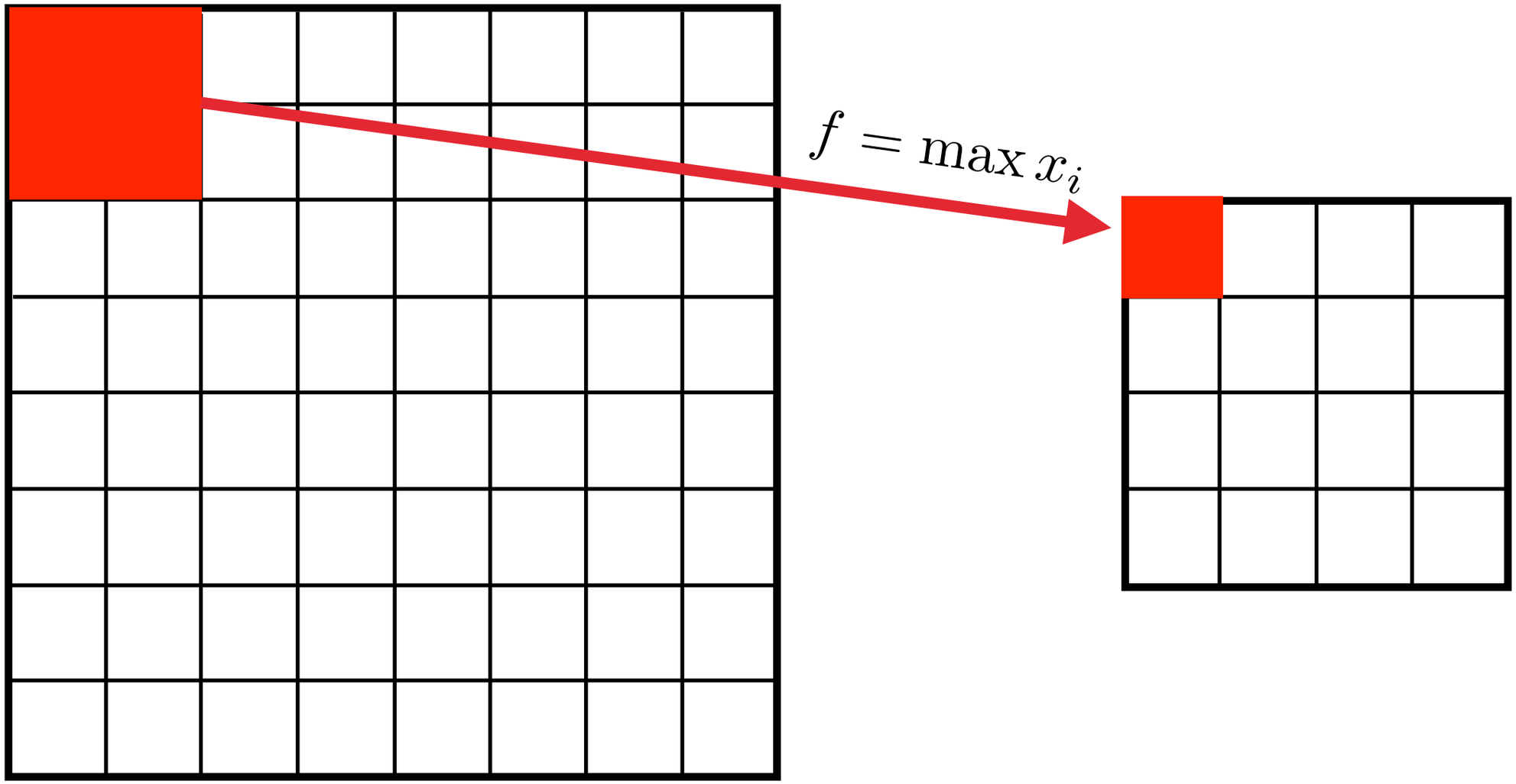
Max pooling is equivalent but taking

$$f = \max x_i$$

The output is a smaller image.

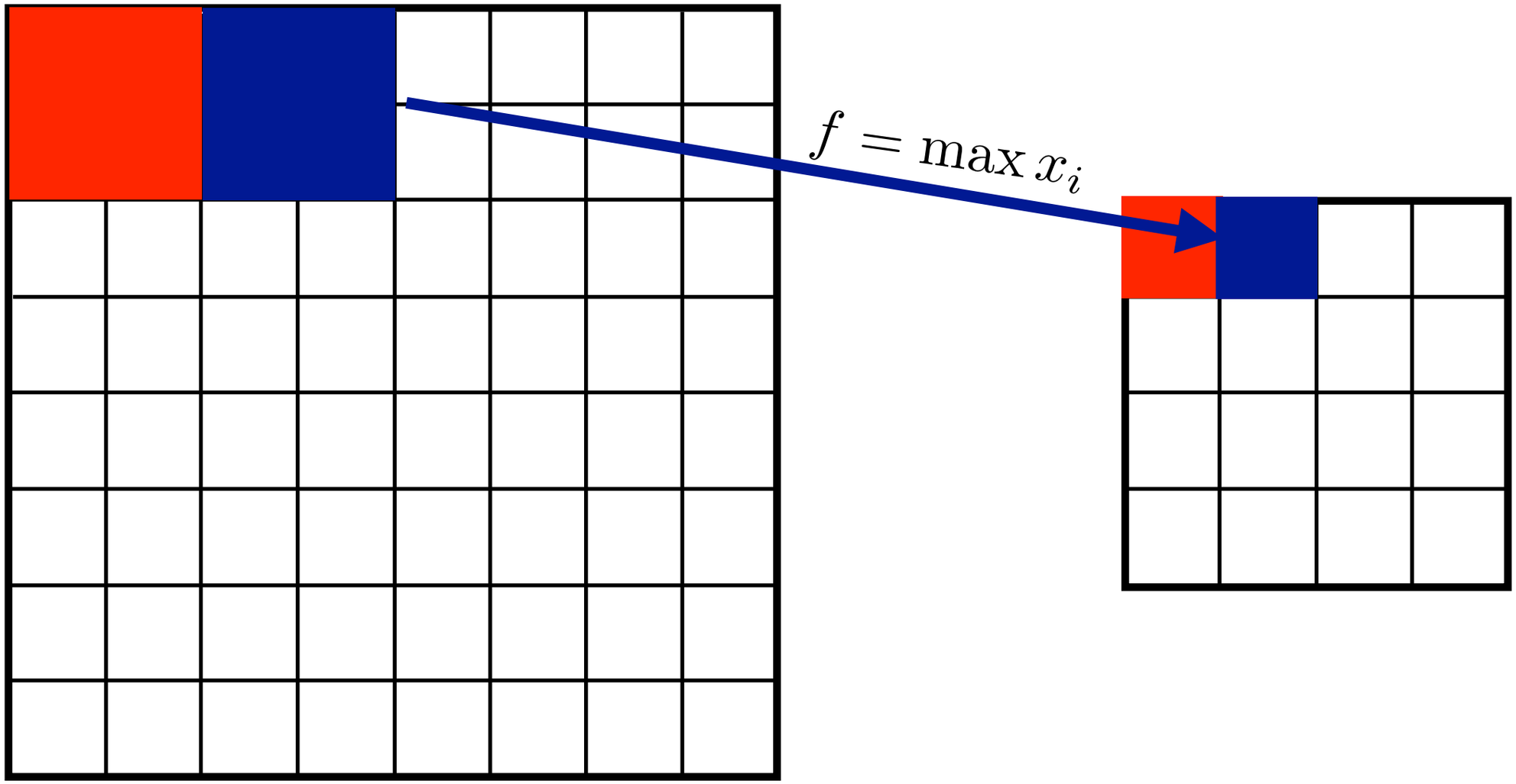
POOLING

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



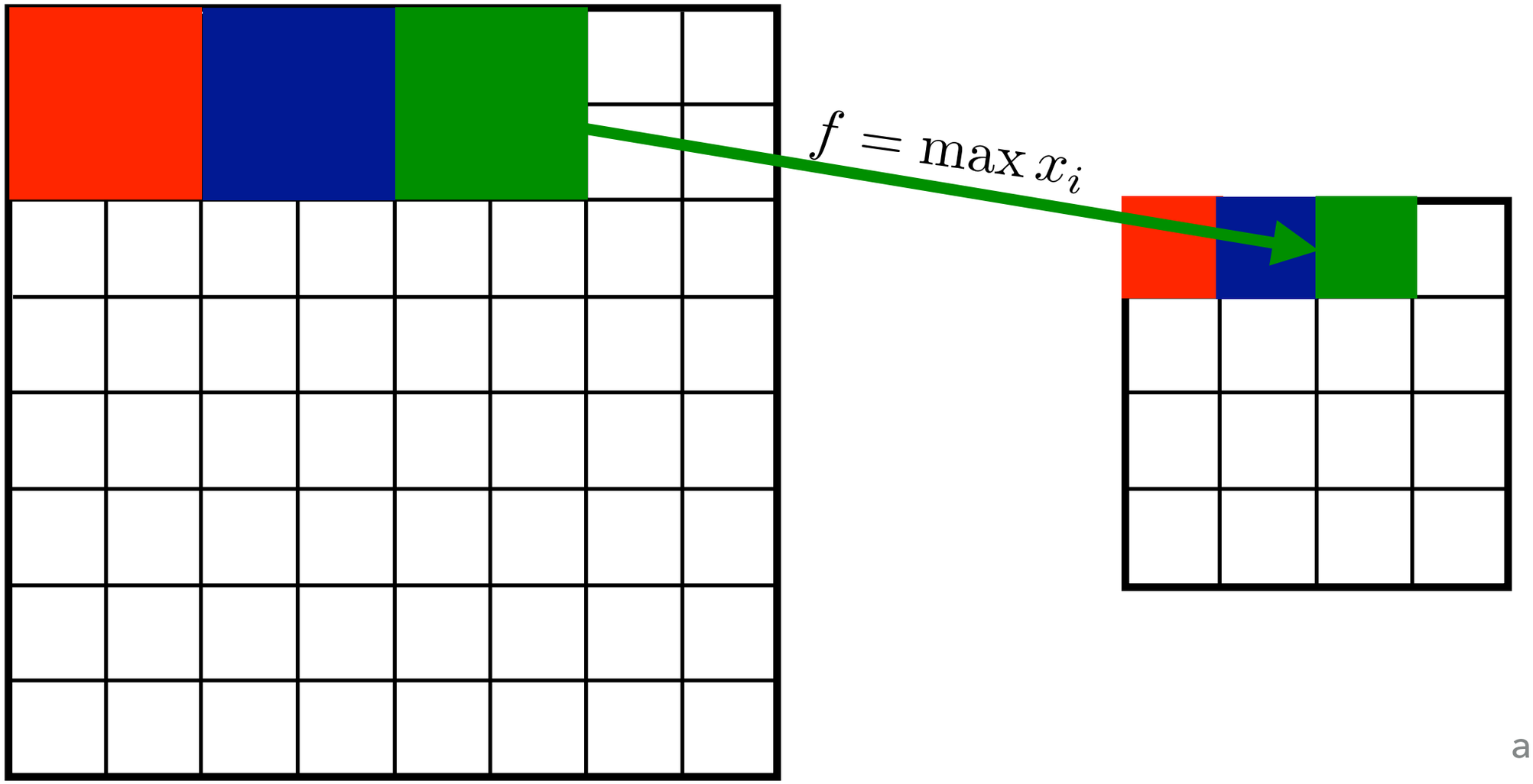
POOLING

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



POOLING

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



and so on...

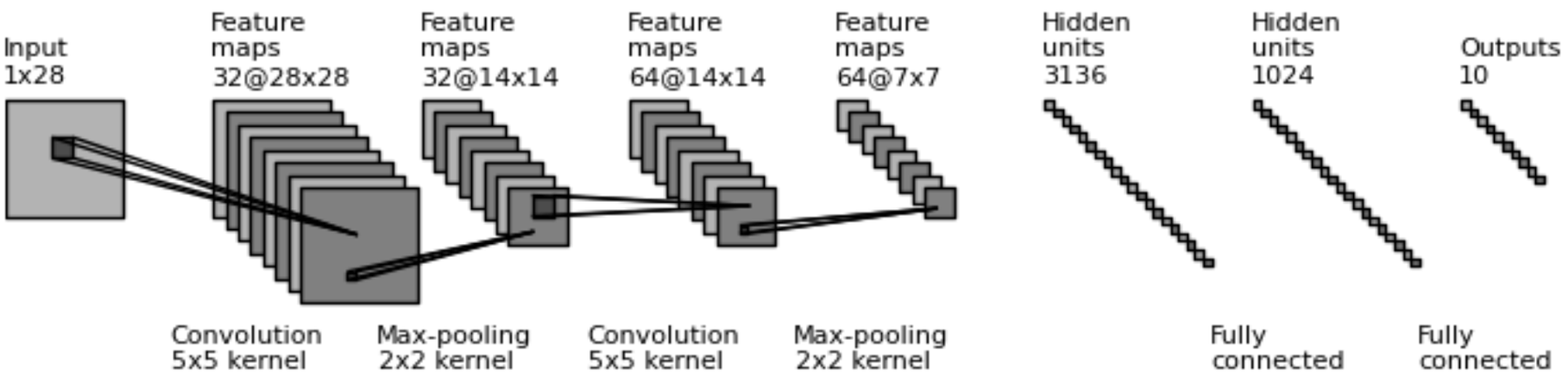
CONVOLUTIONAL NEURAL NETWORK (CNN) ARCHITECTURES

- 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.

CONVOLUTIONAL NEURAL NETWORK (CNN) ARCHITECTURES

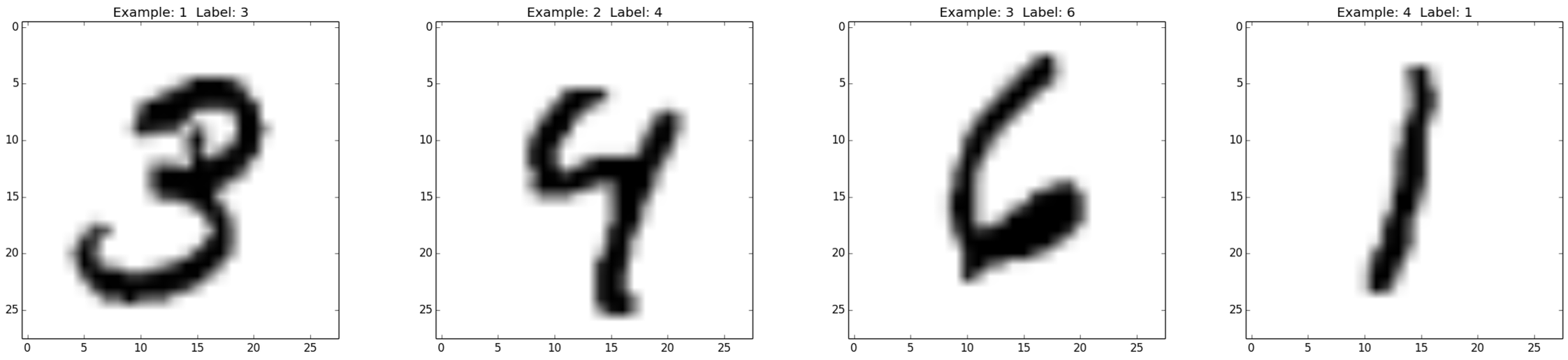
- We can include multiple convolution layers layers that may add to the information extracted from the image.
- We can add pooling layers to reduce the dimensionality.
- e.g. MNIST: handwritten numbers from 0 to 9.



- 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.

CONVOLUTIONAL NEURAL NETWORK (CNN) 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/

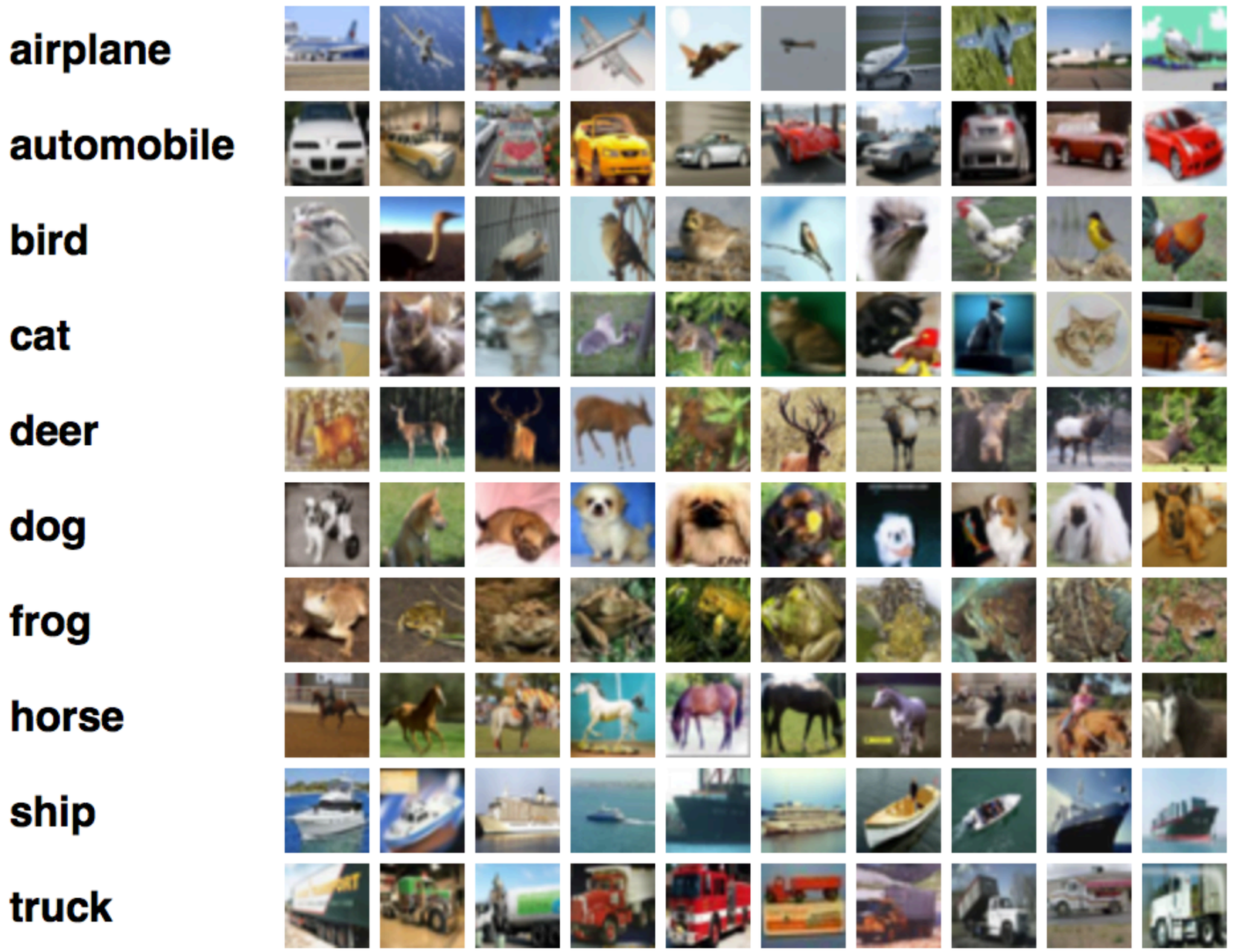
CONVOLUTIONAL NEURAL NETWORK (CNN) ARCHITECTURES/INPUT DATA

- So far we have focussed on monochrome images with a single number 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

CONVOLUTIONAL NEURAL NETWORK (CNN) ARCHITECTURES/INPUT DATA

- CFAR 10 examples:



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

CONVOLUTIONAL NEURAL NETWORK (CNN) ARCHITECTURES/INPUT DATA

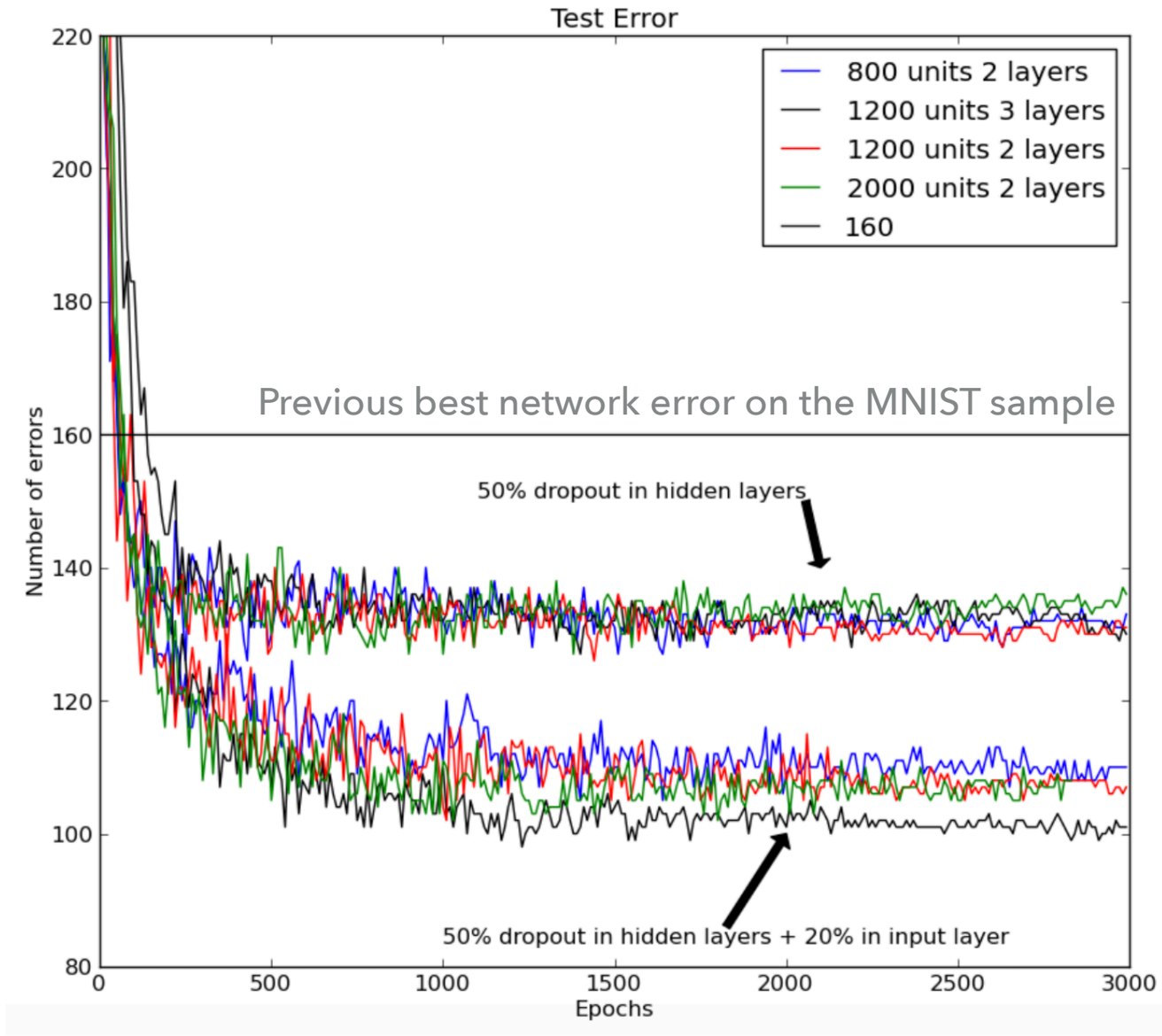
- More abstract problems can be addressed in the same way.
- Some examples are given below:
  - Transient searches can be addressed by stacking images together to form an image of depth D.
  - Tracking problems can be addressed by stacking measurement data from successive layers.
  - More arbitrary problems 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.

DROPOUT[1]

- A problem with deep networks is the number of parameters that need to be determined.
- This leads to a requirement for very large data sets to avoid overfitting (or fine-tuning).
- Dropout is a way to mitigate overfitting (it does not guarantee that there will be no over fitting, just promotes generalised model parameter selection).
  - Randomly remove some fraction of nodes in a network for a given training epoch.
  - Weights are determined on a sample of the events used for training to reduce the susceptibility to honing in on statistical fluctuations.

[1] G. Hinton et al. arXiv:1207.0580

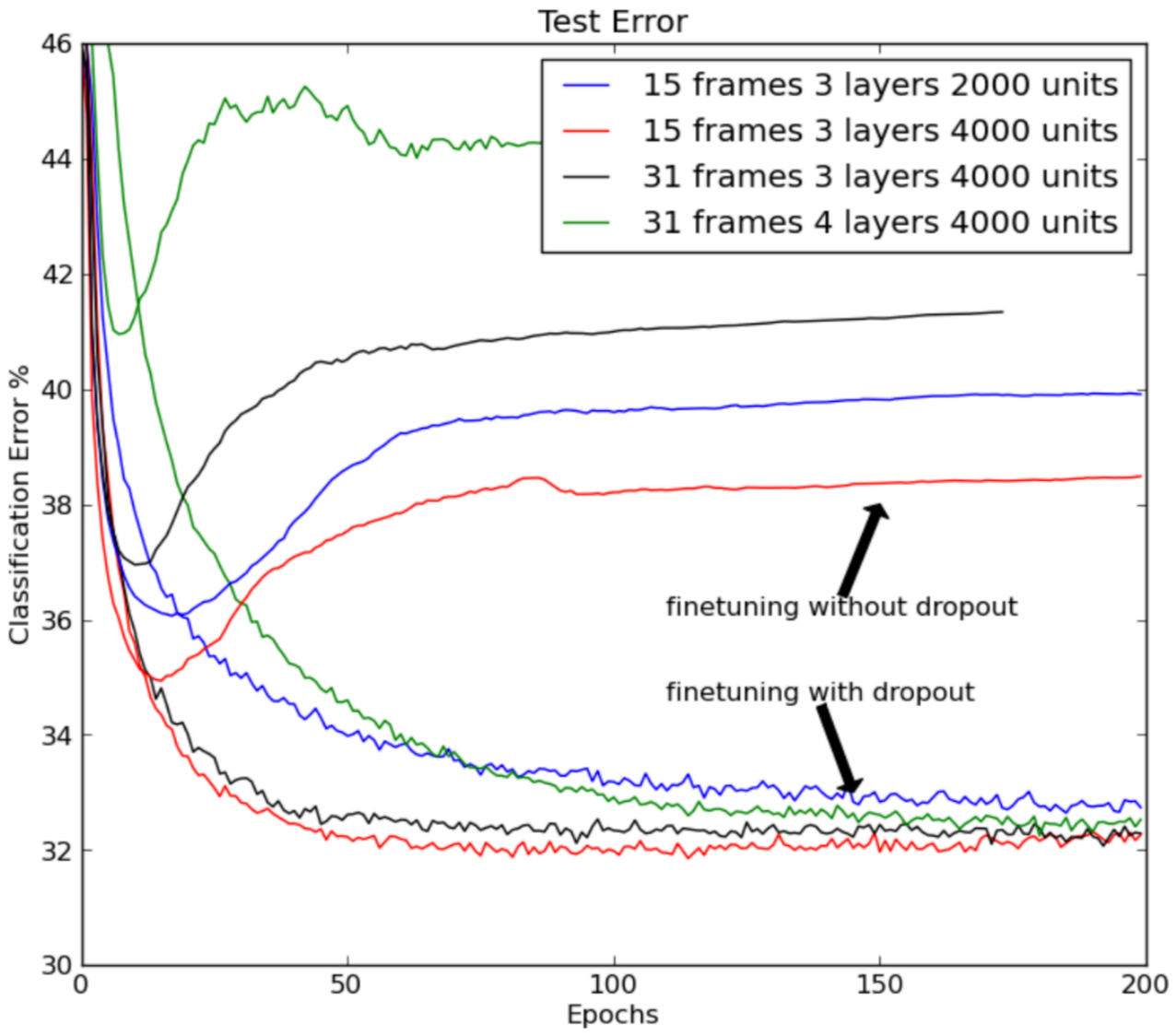
DROPOUT[1]



- Example from Hinton et al. for an MNIST sample.
- Using 50% drop out on the hidden layers gives an improvement over the previous best network architecture.
- Adding 20% drop out in the input layer provides further improvement in error.

[1] G. Hinton et al. arXiv:1207.0580

DROPOUT[1]



- Example from Hinton et al. for a voice recognition sample of data (TIMIT).
- Illustrates the classification error as a function of epoch with and without dropout.

[1] G. Hinton et al. arXiv:1207.0580

SUMMARY

- Discussed the process of convolution of images, and the need for padding.
- The pooling concept: max-pooling and average pooling were discussed.
- The basic building blocks of CNN architectures have been combined together to build an example CNN.
- Dropout has been discussed in the context of parameter training.

SUGGESTED READING

- MNIST: http://yann.lecun.com/exdb/mnist/
- CFAR10: https://www.cs.toronto.edu/~kriz/cifar.html
- Other types of CNN model can be found in the literature, including
  - AlexNet: 5 conv layers and 3 fully connected layers
    - https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks
  - Inception Models: (there are several variations of inception model)
    - e.g. see https://www.cs.unc.edu/~wliu/papers/GoogLeNet.pdf

SUGGESTED READING

- Discussion of event classification in text books
  - Goodfellow: *Deep Learning*
    - Section: II, chapter 9

NOTEBOOK FOR TODAY

https://github.com/abbeywaldron/cinvestav_ML_2024