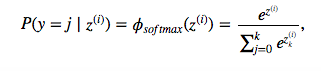
Study notes:

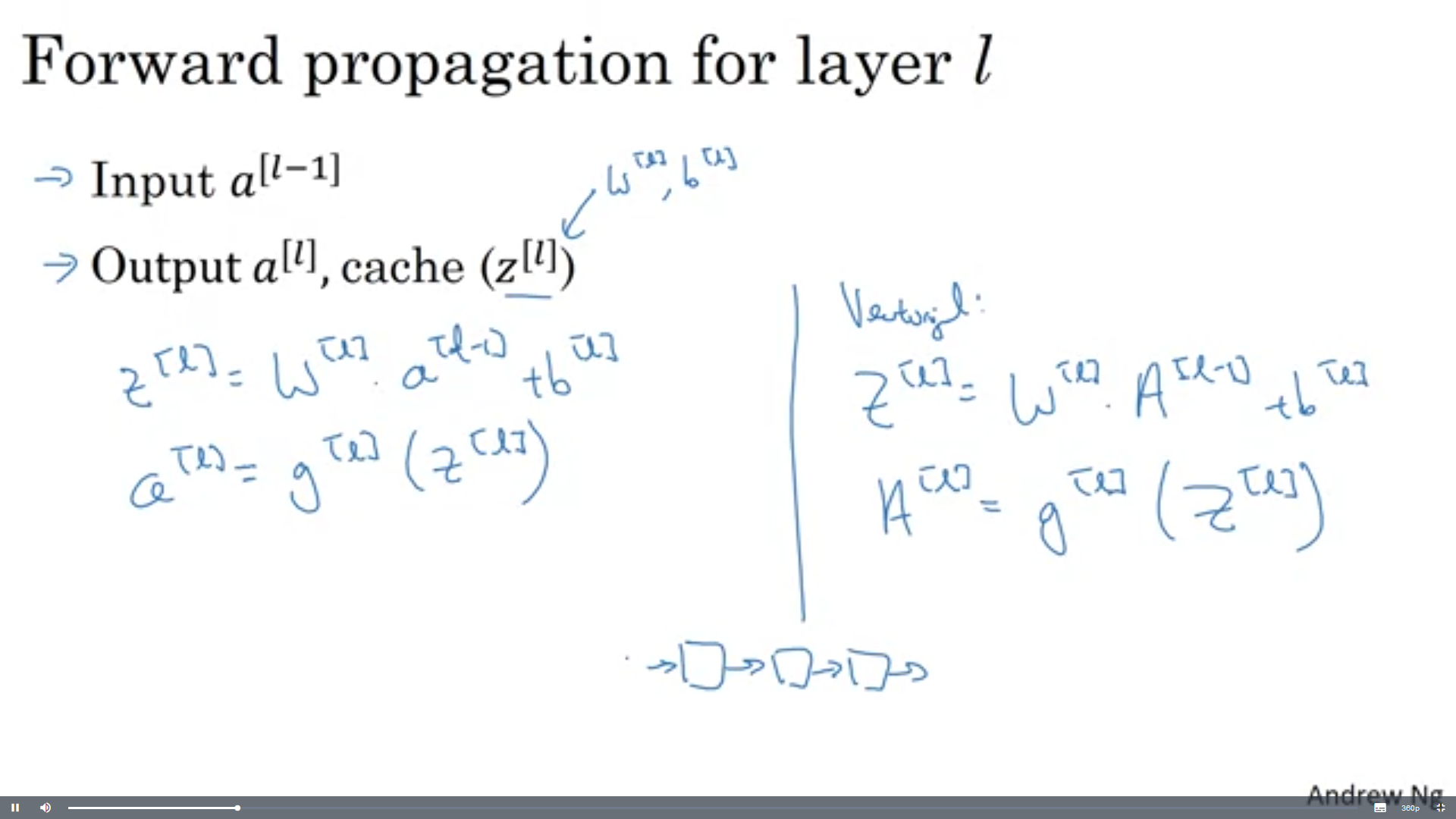
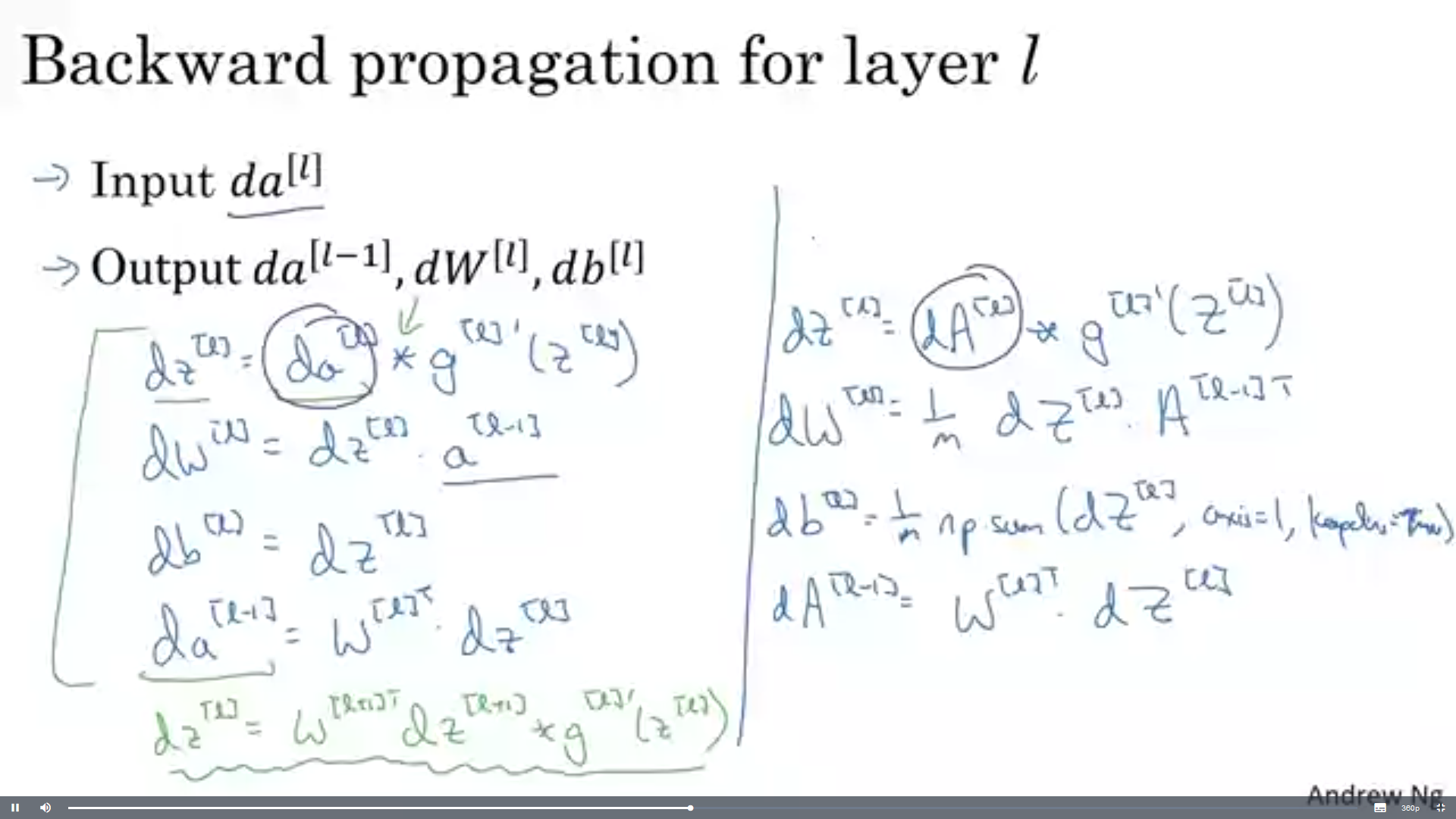
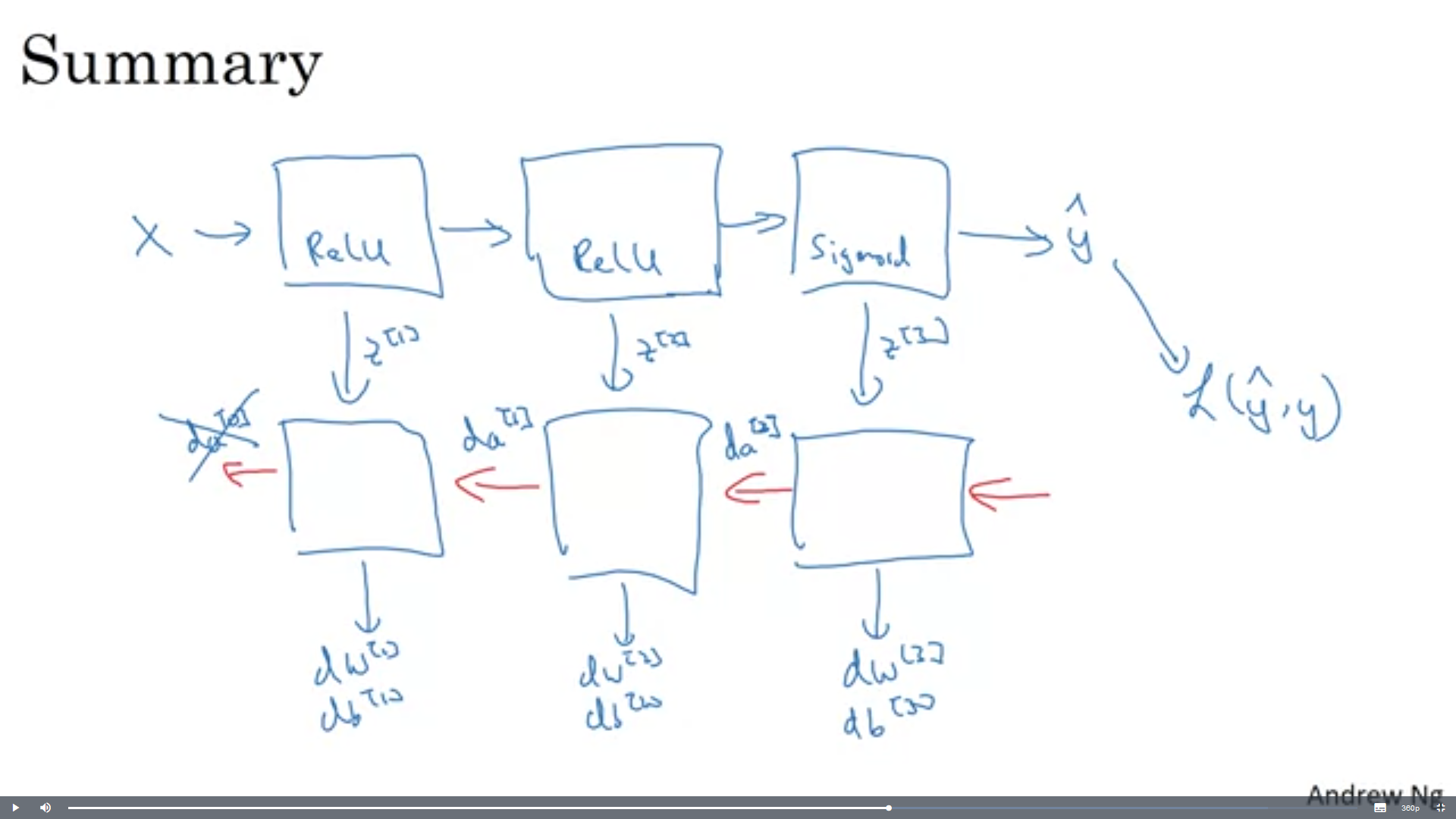
Lecture 01

* Computer vision relates to – neuroscience 40%, cognitive sciences 60%, (graphics, algorithms, system, theory, …) 10%, information retrieval 10%, machine learning 90%, image processing 10%, robotics 10%
* Ways to detect an object in an image
  + Template matchine – naïve approach. Only works well when the image is exactly the same
  + Bag-of-words representation – map pixels into a more robust form, then compare those mapped forms, finally, select the closest image map
    - Computer edges, compute color histograms, gradients, HOG, SIFT, …
    - Uses edges helps to identify shapes. You also have a map of how many times that edge shows up in the image
* Face recognition was a breakthrough technology 15 years ago

Lecture 02

* Linear classifier
  + A straight line that
* Logistic regression
  + Is a single neuron, a neural network goes back to update as it learns more
  + Used to solve the classification problem
* Loss function for logistic regression
  + Calculates how many of the tests are classified incorrectly
* Cost function for logistic regression
  + Tells how good the algorithm is currently doing ( at the current iteration)
* Gradient decent algorithm
* Computation graph
* Derivatives for logistic regression
* ~~Implementing logistic regression in python~~
* Sigmoid
  + Takes any parameter and returns a value between 0 and 1
  + Good for neurons with binary classifiers
  + Is used at the end
* Handling multiple classes in neural networks more than isCat notCat
* One vs all
  + You run binary classifier on all classes and take the one with the highest likelihood
* Softmax regression
  + Define a new cost function and derive all the weight update rules according to that cost function
  + Is a replacement for sigmoid. Use the following instead of sigmoid
  + 
* ReLU
  + a = g(z) = max(0,z)
  + used in the intermediate stages
* know the derivatives for the activation functions
* Neural networks
* Fully connected (FC) Neural Network
  + Means that there is a weight for every parameter in the network
  + Learn how to calculate dimensions for W
  + All neurons are connected to each other

Lecture 03/4

* Deep vs shallow
* Deep NN notation
* Hyperparameters
  + **Hyperparameters** are the **variables which determines the network structure**(Eg: Number of Hidden Units) and the **variables which determine how the network is trained**(Eg: Learning Rate).
* Forward vs backward computation
  + Forward step – loss function, functions computing prediction, cost, then gradients
    - Continuously computing chain rule
  + 
  + 
  + 
  + When doing forward propagation, you would save your cache as you go
  + Then compute the activation function and feed into the next layer
  + Finally getting the output, then computing the loss function
  + Feed back the loss function and relate the cache to the loss function and recalculate
* Backward propagation
* Convolutional neural networks
  + Features
  + Edge detection
  + Edges
  + Edge models
  + Characterizing edges
  + Derivatives and average
    - Discrete derivative
    - Finite difference
    - Derivative in 2-d
    - Derivative of images
* A CNN consists of an input and an output layer, as well as multiple [hidden layers](https://en.wikipedia.org/wiki/Multilayer_perceptron#Layers). The hidden layers of a CNN typically consist of convolutional layers, pooling layers, fully connected layers and normalization layers[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)].
* Description of the process as a [convolution](https://en.wikipedia.org/wiki/Convolution) in neural networks is by convention. Mathematically it is a [cross-correlation](https://en.wikipedia.org/wiki/Cross-correlation) rather than a convolution. This only has significance for the indices in the matrix, and thus which weights are placed at which index.

### Convolutional

* Convolutional layers apply a convolution operation to the input, passing the result to the next layer. The convolution emulates the response of an individual neuron to visual stimuli.[[7]](https://en.wikipedia.org/wiki/Convolutional_neural_network#cite_note-deeplearning-7)
* Each convolutional neuron processes data only for its [receptive field](https://en.wikipedia.org/wiki/Receptive_field).
* Although [fully connected feedforward neural networks](https://en.wikipedia.org/wiki/Multilayer_perceptron) can be used to learn features as well as classify data, it is not practical to apply this architecture to images. A very high number of neurons would be necessary, even in a shallow (opposite of deep) architecture, due to the very large input sizes associated with images, where each pixel is a relevant variable. For instance, a fully connected layer for a (small) image of size 100 x 100 has 10000 weights for *each* neuron in the second layer. The convolution operation brings a solution to this problem as it reduces the number of free parameters, allowing the network to be deeper with fewer parameters.[[8]](https://en.wikipedia.org/wiki/Convolutional_neural_network#cite_note-8) For instance, regardless of image size, tiling regions of size 5 x 5, each with the same shared weights, requires only 25 learnable parameters. In this way, it resolves the vanishing or exploding gradients problem in training traditional multi-layer neural networks with many layers by using [backpropagation](https://en.wikipedia.org/wiki/Backpropagation)[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)].

### Pooling

* Convolutional networks may include local or global pooling layers[[*clarification needed*](https://en.wikipedia.org/wiki/Wikipedia:Please_clarify)], which combine the outputs of neuron clusters at one layer into a single neuron in the next layer.[[9]](https://en.wikipedia.org/wiki/Convolutional_neural_network#cite_note-flexible-9)[[10]](https://en.wikipedia.org/wiki/Convolutional_neural_network#cite_note-10) For example, *max pooling* uses the maximum value from each of a cluster of neurons at the prior layer.[[11]](https://en.wikipedia.org/wiki/Convolutional_neural_network#cite_note-mcdns-11) Another example is *average pooling*, which uses the average value from each of a cluster of neurons at the prior layer[[*citation needed*](https://en.wikipedia.org/wiki/Wikipedia:Citation_needed)].

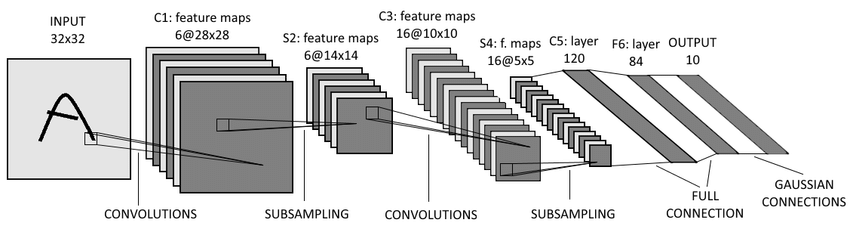
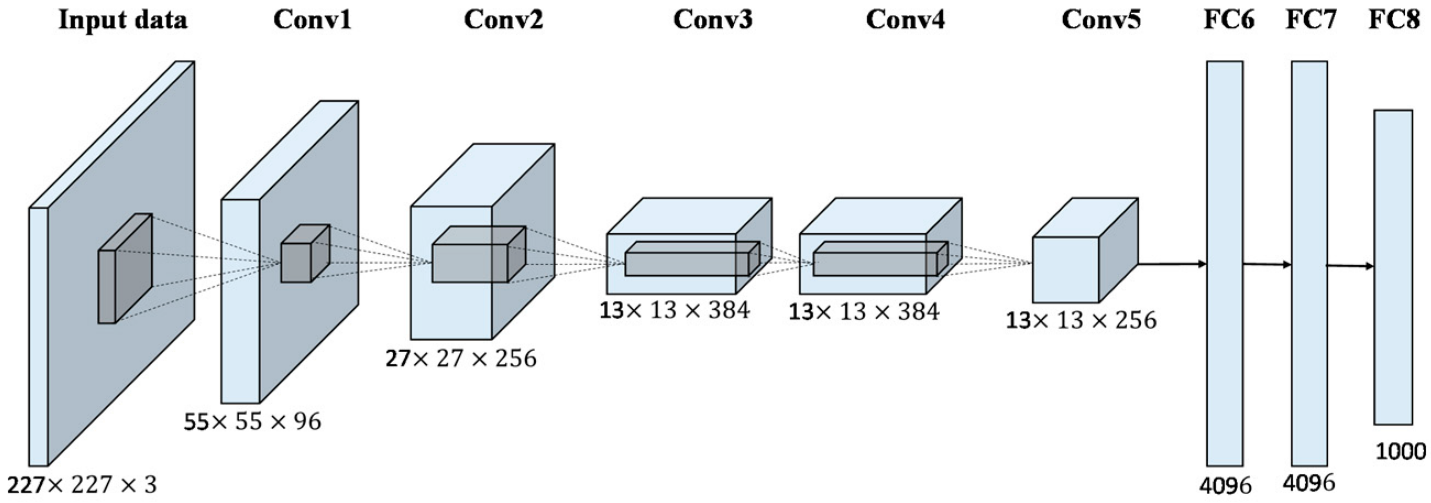
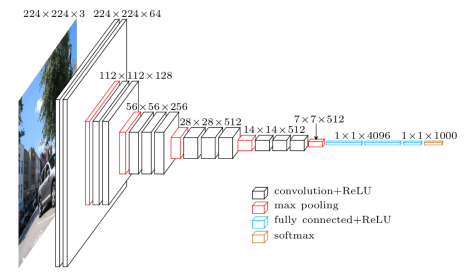
### Fully connected

* Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural network ([MLP](https://en.wikipedia.org/wiki/Multilayer_perceptron)).

### Receptive field

* In neural networks, each neuron receives input from some number of locations in the previous layer. In a fully connected layer, each neuron receives input from *every* element of the previous layer. In a convolutional layer, neurons receive input from only a restricted subarea of the previous layer. Typically the subarea is of a square shape (e.g., size 5 by 5). The input area of a neuron is called its *receptive field*. So, in a fully connected layer, the receptive field is the entire previous layer. In a convolutional layer, the receptive area is smaller than the entire previous layer.

### Weights

* Each neuron in a neural network computes an output value by applying some function to the input values coming from the receptive field in the previous layer. The function that is applied to the input values is specified by a vector of weights and a bias (typically real numbers). Learning in a neural network progresses by making incremental adjustments to the biases and weights. The vector of weights and the bias are called a *filter* and represents some feature of the input (e.g., a particular shape). A distinguishing feature of CNNs is that many neurons share the same filter. This reduces memory footprint because a single bias and a single vector of weights is used across all receptive fields sharing that filter, rather than each receptive field having its own bias and vector of weights. [[1]](https://en.wikipedia.org/wiki/Convolutional_neural_network#cite_note-LeCun-1)
* Vertical edge detection
* Horizontal edge detection
* Sobel filter
  + creates an image emphasising edges
* Scharr filter
  + function computes the gradients of input image in both x and y direction by convolving the kernel with input image being processed.
* Using convolution to pick a filter
* Padding
* Convolutions on an RGB image
* Multiple filters for RGB image
* ConvNet
  + A convolutional layer of CNN
  + Sequence of convolution layers interspersed with activation functions
* Know parameters you’ll have in a conv layer with 10 filters that are 3x3x20 dimensional each
* ConvNet common layers
  + Learned layers
    - Convolutional layer
      * Is just the dot product of several pixels to make into a smaller size
    - Fully connected
    - Locally connected
  + No learned parameters layers
    - Pooling
    - Activation layer
* Pooling layer
  + Max pooling
  + Average pooling
* Activation map
  + If there were 6 5x5 filters then there would need to be 6 separate activation maps
* LeNet-5
  + 
* Why should a ConvNet be deep?
* AlexNet
  + 
* VGG – 16
  + 
* Residual Network
  + A network that learns to skp over layers
  + Solves the problem of vanishing gradients by reusing activation from a previous layer until the layer next to the current one have learned its weights
* Residual building block
* vanishing gradient problem
  + is a difficulty found in training artificial neural networks with gradient-based learning methods and backpropagation. In such methods, each of the neural network's weights receives an update proportional to the partial derivative of the error function with respect to the current weight in each iteration of training. The problem is that in some cases, the gradient will be vanishingly small, effectively preventing the weight from changing its value. In the worst case, this may completely stop the neural network from further training.k
* ResNet
* 1x1 convolution
  + leads to dimension reductionality
* Inception network
  + A network that gets wider instead of deeper
  + Do a number of filters on the same level, then concatenate them and send them to the next module
  + https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202

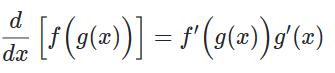
Lecture 05

* Convolutions on an RGB image
* Calculate the input/output layers for an image in a ConvNet based on padding size, stride size, and filter size

Lecture 6

* Inception net and 1x1 convolution
  + A 1x1 convolution can shrink parameters or increase based on the size of the filter parameter and whether it’s bigger than the previous one or not
  + 1x1 convolutions are used to compute reducctions before the expensive 3x3 and 5x5 convolutions
  + Also include the use of rectified linear activation
* Classification and localization
  + Is there a turtle in this picture? If yes, localize
* Regression head
* Classification head
* Detection as regression
* Region proposals
* Selective search
* R-CNN (Regions with CNN features)

Other

* Gradient decent optimization
  + Gradient is another word for slope
  + Relates how variables relate to each other
* Hidden layer3.
* Activation
  + Activate at each level of the neural network
  + So a linear line can bend to make the best classification
* Momentum
  + Similar to learning rate in gradient algorithms
  + Momentum helps to know the direction of the next step with the knowledge of the previous steps. It helps to prevent oscillations. A typical choice of momentum is between 0.5 to 0.9.
* Overfitting
  + When there’s too much training and the classifier line gets too close to the parameters
  + This can cause errors if you don’t have enough data since the ones near the line may go in the wrong direction
* Chain rule
  + 
* Transpose of a matrix
  + Flips a matrix over its diagonal.
  + <https://en.wikipedia.org/wiki/Transpose>
* Numpy reshape
  + numpy allow us to give one of new shape parameter as -1 (eg: (2,-1) or (-1,3) but not (-1, -1)). It simply means that it is an unknown dimension and we want numpy to figure it out. And numpy will figure this by looking at the 'length of the array and remaining dimensions' and making sure it satisfies the above mentioned criteria