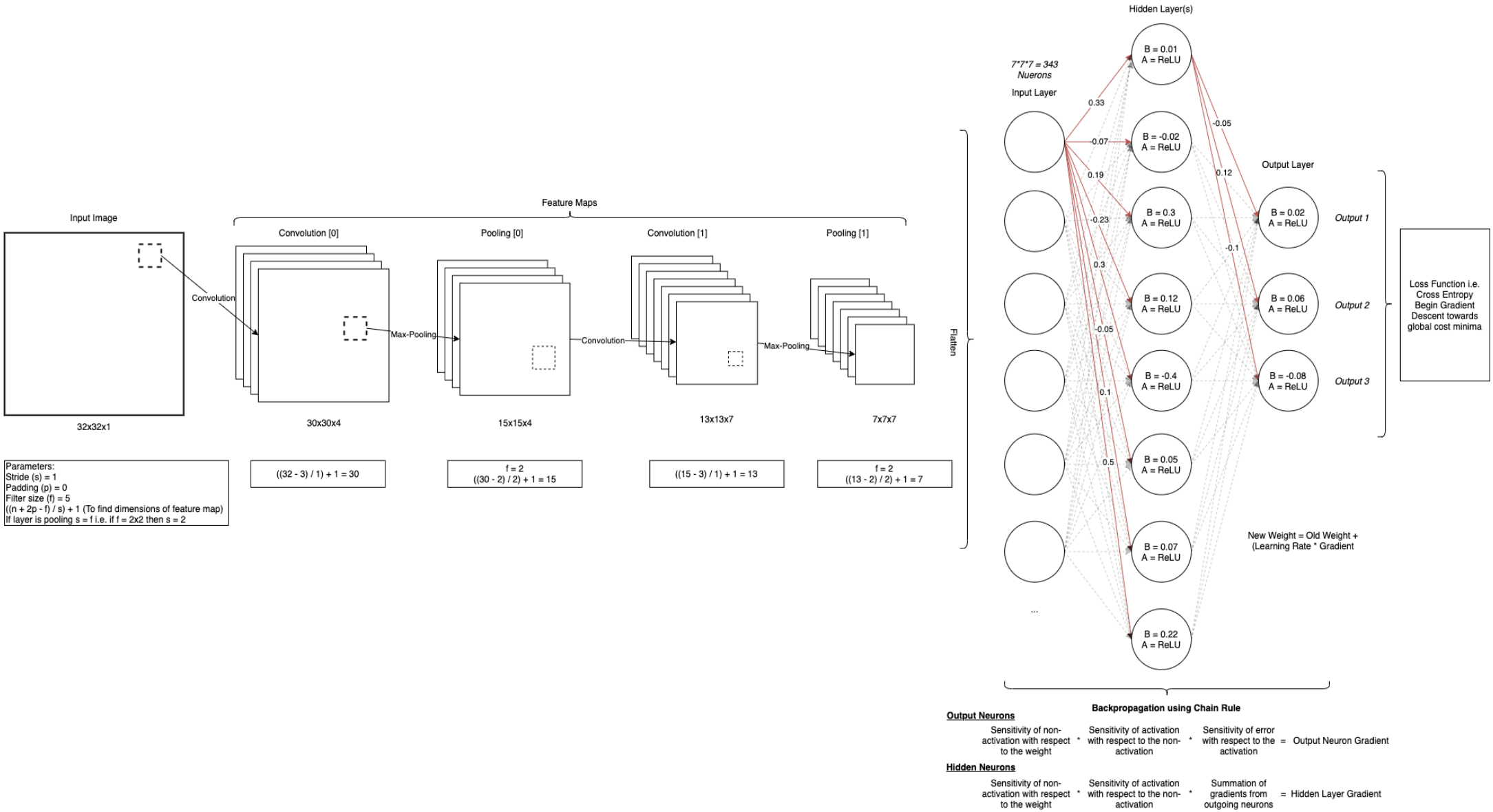
Feature Maps
Input Image
Convolution [0]
Pooling [0]
Convolution [1]
Pooling [1]
Convolution
Max-Pooling
Convolution
Max-Pooling
32x32x1
30x30x4
15x15x4
13x13x7
7x7x7
Parameters:
Stride (s) = 1
Padding (p) = 0
Filter size (f) = 5
((n + 2p - f) / s) + 1 (To find dimensions of feature map)
If layer is pooling s = f i.e. if f = 2x2 then s = 2
((32 - 3) / 1) + 1 = 30
f = 2
((30 - 2) / 2) + 1 = 15
((15 - 3) / 1) + 1 = 13
f = 2
((13 - 2) / 2) + 1 = 7
Flatten
Hidden Layer(s)
7*7*7 = 343 Nuerons
Input Layer
B = 0.01
A = ReLU
B = -0.02
A = ReLU
B = 0.3
A = ReLU
B = 0.12
A = ReLU
B = -0.4
A = ReLU
B = 0.05
A = ReLU
B = 0.07
A = ReLU
B = 0.22
A = ReLU
...
0.33
0.07
0.19
-0.23
0.3
-0.05
0.1
0.5
-0.05
0.12
-0.1
Output Layer
B = 0.02
A = ReLU
B = 0.06
A = ReLU
B = -0.08
A = ReLU
Output 1
Output 2
Output 3
Loss Function i.e. Cross Entropy Begin Gradient Descent towards global cost minima
New Weight = Old Weight + (Learning Rate * Gradient
Backpropagation using Chain Rule
Output Neurons
Sensitivity of non-activation with respect to the weight * Sensitivity of activation with respect to the non-activation * Sensitivity of error with respect to the activation = Output Neuron Gradient
Hidden Neurons
Sensitivity of non-activation with respect to the weight * Sensitivity of activation with respect to the non-activation * Summation of gradients from outgoing neurons = Hidden Layer Gradient

Feature Extraction using Convolution

1) ***Convolution*** performed on input image using filter usually of equal width and height, whereby matrix multiplication is performed in each possible position of the filter in relation to the input image. Though this may be changed using the stride parameter. The output of the convolutions for each filter corresponds directly to a feature map. If there are four filters to be convolved over the image, for example, there are four feature maps. ReLU or other activation functions are often input from the convolutional filter output whereby values <= 0 are assigned 0 and values > 0 are unchanged.
2) ***Pooling*** occurs which may either be max or average pooling. The stride of a pooling layer is usually equal to its filter size. This aids in providing more abstracted representations of feature maps and thus improves the ability to generalise. It also reduces the computational cost of further convolutions, therein reducing the overall dimensions and ultimately reducing the number of input neurons to the classifier.
3) ***Convolution and pooling may occur several consecutive times***. Only reducing the dimensions of feature maps when 'valid padding' is chosen over 'same padding' determining whether to use zero padding for necessity or size preservation. The number of filters and therefore features, are determined when initialising the convolutional layer. These features are automatically extracted without being explicitly defined and are normally highly abstracted.
4) ***Flattened*** to single column vector of size w*h*d whereby each row corresponds directly to one input neuron in the classifier for the image sample.

Classification using "Fully Connected Layer" i.e. The Hidden Layer of a Feedforward Neural Network

1) ***Weight Initialisation*** i.e. Random, Normal Distribution, Xavier, He Norm

2) Forward Propagation occurs

- Computing the non-activation of each neuron
- Non-activation = Summation of (inputs multiplied by weight) + bias
- Computing the activation of each neuron

3) ***The Loss Function calculates the error*** between actual and predicted values

4) ***Using gradient descent, backpropagation occurs***, whereby the objective is to minimise the error by adjusting the weights to reach global minima.

- The amount of times the weights are adjusted depends on minibatch size
- And number of epochs
- Weight Update Rule: New Weight = Old Weight – Learning Rate * Gradient

5) Calculating the gradient for Output Layer Neurons

$$\frac{\partial \mathcal{E}_k}{\partial w_{hj}^k} = \frac{\partial \mathcal{E}_k}{\partial \mathcal{S}(y_j^k)} \frac{\partial \mathcal{S}(y_j^k)}{\partial y_j^k} \frac{\partial y_j^k}{\partial w_{hj}^k}$$

- Requires derivative of Error Function and derivative of Activation Function

6) Calculating the gradient for Hidden Layer Neurons

$$\frac{\partial \mathcal{E}_k}{\partial w_{ih}^k} = \sum_{j=1}^{p} \left\{ \frac{\partial \mathcal{E}_k}{\partial \mathcal{S}(y_j^k)} \frac{\partial \mathcal{S}(y_j^k)}{\partial y_j^k} \frac{\partial y_j^k}{\partial \mathcal{S}(z_h^k)} \right\} \mathcal{S}'(z_h^k) \mathcal{S}(x_i^k)$$

Optimisations for Feature Extraction using Convolution

Convolution and Pooling Architectural Design - Convolutional Depth determines number of parameters to Fully Connected Layers - Sequencing of Convolutional and Pooling Layers - Filter Depth i.e. Number of filters/features to be extracted, dimensions, stride, same padding or valid padding
Data Augmentation Techniques; Provides more data - Cut-Out, Random Brightness, Zoom, Flip, Shift, Merging Images, Noise Added
Regularisation; Reduce Overfitting, Increase Generalisation – Weight Decay (*L2,* L1), Early Stopping, Dropout
Optimisation; Momentum, batch-norm (counteracts internal covariate shift), learning rate scheduling, AdaGrad, RMSProp, Adam
Epochs: Using more reduces error more though there is a vanishing gradient
Using Pre-Trained Models: Save time and just change the output layer of a pretrained CNN i.e. AlexNet, LeNet, ResNet, VGGNet, Inception
Initialisation: Weight and bias initialisation