**Computer vision and machine learning interview questions:**

**Questions + Answers are available in this link given below, follow the link below to learn more:**

<https://github.com/andrewekhalel/MLQuestions>

<https://github.com/theainerd/MLInterview>

<https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-interview-questions> (Deep learning interview questions)

<https://www.analyticsvidhya.com/blog/2020/04/comprehensive-popular-deep-learning-interview-questions-answers/> (Deep learning interview questions)

<https://towardsdatascience.com/50-deep-learning-interview-questions-part-1-2-8bbc8a00ec61> (Deep learning interview questions)

**### 30 days of interview praparation from krish naik:**

<https://github.com/iNeuronai/interview-question-data-science->

Youtube: Image filtering in python tutorials by Apeer\_micro

**=====================================================================================**

**Que1.** **What is linear and logistic regression? Difference between them?**

See this for ML: ([Machine Learning Course for Beginners](https://www.youtube.com/watch?v=NWONeJKn6kc&ab_channel=freeCodeCamp.org)



)

**Ans:** See link: <https://dhirajkumarblog.medium.com/top-5-difference-between-linear-regression-and-logistic-regression-893f6470d7e0>

<https://www.javatpoint.com/linear-regression-vs-logistic-regression-in-machine-learning>

Linear regression well explained:

================================================================

<https://www.scribbr.com/statistics/simple-linear-regression/>

<https://towardsdatascience.com/linear-regression-with-example-8daf6205bd49>

<https://www.kdnuggets.com/2020/03/linear-logistic-regression-explained.html>

**=====================================================================================**

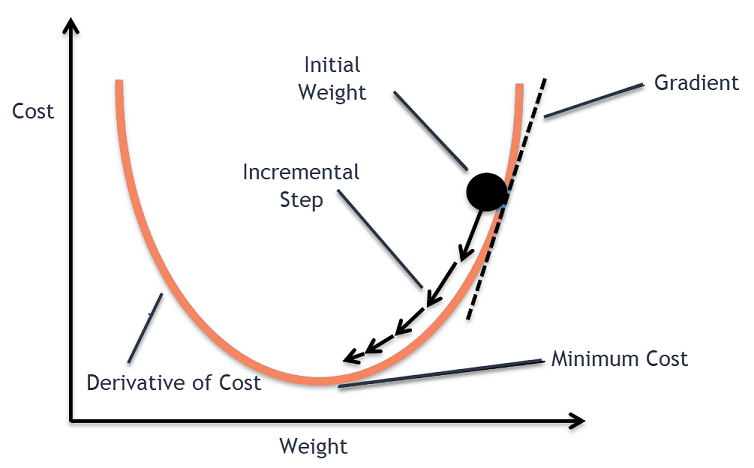
**Que2. What is Gradient descent and types of gradient descent?**

**Ans:** <https://www.analyticsvidhya.com/blog/2020/10/how-does-the-gradient-descent-algorithm-work-in-machine-learning/>

<https://towardsdatascience.com/batch-mini-batch-stochastic-gradient-descent-7a62ecba642a>

And

<https://builtin.com/data-science/gradient-descent>



# **Batch Gradient Descent:**

# In Batch Gradient Descent, all the training data is taken into consideration to take a single step. We take the average of the gradients of all the training examples and then use that mean gradient to update our parameters.

# So that’s just one step of gradient descent in one epoch.

* Batch Gradient Descent is great for convex or relatively smooth error manifolds (convex error manifold is having on global minima I.e., no local minima).

**Stochastic gradient descent:**

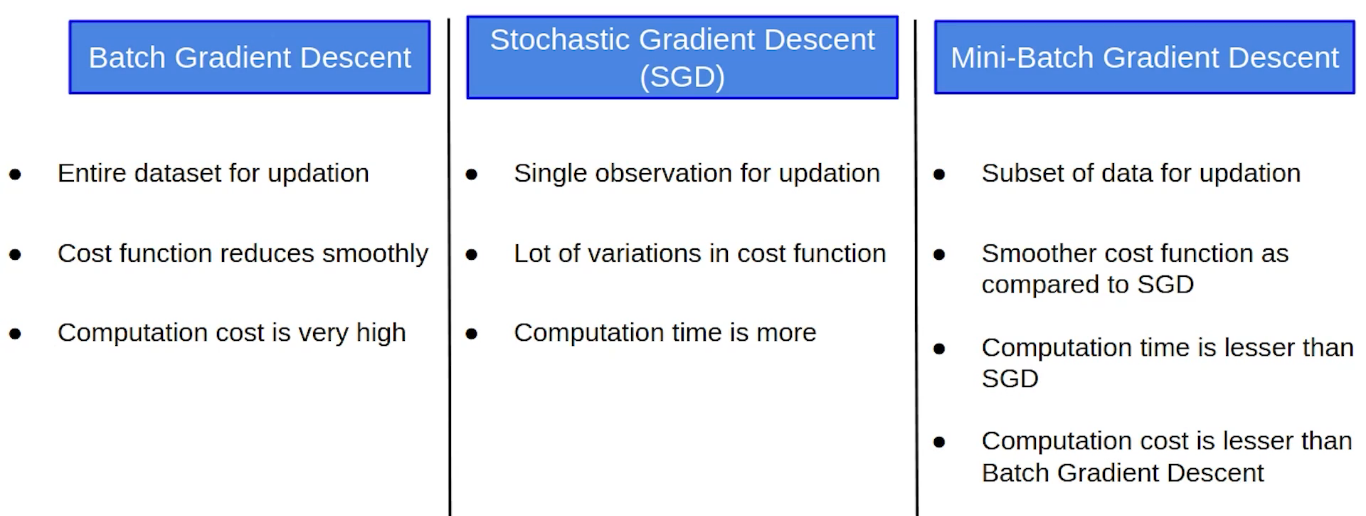
* In Stochastic Gradient Descent (SGD), we consider just one example at a time to take a single step.
* SGD can be used for larger datasets. It converges faster when the dataset is large as it causes updates to the parameters more frequently.
* Since we are considering just one example at a time the cost will fluctuate over the training examples and it will **not** necessarily decrease. But in the long run, you will see the cost decreasing with fluctuations.



**Mini-batch gradient descent:**

* Neither do we use all the dataset all at once nor we use the single example at a time.
* We use a batch of a fixed number of training examples which is less than the actual dataset and call it a mini-batch

Types of gradient descent: <https://www.analyticsvidhya.com/blog/2021/03/variants-of-gradient-descent-algorithm/>



**=====================================================================================**

**Que3. What is ROC and AUC? And what do you mean by Type I and Type II errors? What is sensitivity, specificity, FPR, and relations between them?**

**Ans:** See this link <https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5>

**=====================================================================================**

**Que4. What is Precision-Recall curve? How is it different from ROC curve?**

**Ans:** See this link <https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-classification-in-python/>

* ROC Curves summarize the trade-off between the true positive rate and false positive rate for a predictive or classification model using different probability thresholds.
* Precision-Recall curves summarize the trade-off between the true positive rate and the positive predictive value for a predictive model using different probability thresholds.
* **ROC curves are appropriate when the observations are balanced between each class, whereas precision-recall curves are appropriate for imbalanced datasets.**

**=====================================================================================**

**Que5. What is feature scaling? Explain data normalization and data standardization?**

**Ans:**

**=====================================================================================**

**Que6. Bias and variance trade-off? What is underfitting, overfitting (statistical fitting), how to combat underfitting and over fitting?**

**Ans:**

For exact bias and variance trade-off check this below article which explains the same with **Bull’s eye diagram:** <https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229>

**Underfitting:**

* + High Bias and low variance.
  + When a model fails to capture important distinctions and patterns in the data, it performs poorly even in training data, that is called underfitting.

**Overfitting:**

* + Low Bias and High Variance.
  + overfitting, where a model matches the training data almost perfectly, but does poorly in validation and other new data



Techniques to reduce underfitting:

* + Increase model complexity
  + Increase number of features, performing feature engineering
  + Remove noise from the data.
  + Increase the number of epochs or increase the duration of training to get better results.

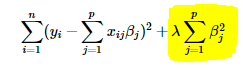
Techniques to reduce overfitting:

* + Increase training data.
  + Reduce model complexity.
  + Early stopping during the training phase (have an eye over the loss over the training period as soon as loss begins to increase stop training).
  + Use Regularization techniques such as Ridge Regularization (L2) and Lasso Regularization (L1).
  + Use dropout for neural networks to tackle overfitting.

**=====================================================================================**

**Que7.**What is Regularization? What are their types? Explain L1, L2 regularization?

**Ans:** <https://towardsdatascience.com/l1/-and-l2-regularization-methods-ce25e7fc831c>

* **Regularization** is a technique to prevent overfitting by penalizing the coefficients of the cost function as model complexity increases.
* when you have a large number of features in your dataset, some of the Regularization techniques are used.
* A regression model that uses L1 regularization technique is called ***Lasso Regression*** and model which uses L2 is called ***Ridge Regression***.
* *The key difference between these two is the* ***penalty term which is added to loss function****.*
* **Ridge regression** adds “*squared magnitude*” of coefficient as penalty term to the loss function.
* 
* **Lasso Regression** (Least Absolute Shrinkage and Selection Operator) adds “*absolute value of magnitude*” of coefficient as penalty term to the loss function.
* 
* The **key difference** between these techniques is that **Lasso (L1) shrinks the less important feature’s coefficient to zero** thus, removing some feature altogether. So, this works well for **feature selection** in case we have a huge number of features.

**=====================================================================================**

**Que8: What are activation functions? Why are they used (to activate neurons)? Types of activation functions?**

**Ans:** <https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-learning-models-84ba9f82c253#:~:text=Simply%20put%2C%20an%20activation%20function,fired%20to%20the%20next%20neuron>.

**=====================================================================================**

**Que9:** what is epoch, batch and iteration in model training?

**Ans:**

**Epoch:** one forward pass and one backward pass of **all** the training examples

**Batch:** examples processed together in one pass (forward and backward)

**Iteration:** number of training examples / Batch size

**=====================================================================================**

**Que10:** What is quantization? Why is it used? **(Interview question at Starkenn technologies)** OR how we can reduce runtime/inference time of edge devices while maintaining the same accuracy of the model?

**Ans**: <https://scaledown-team.github.io/blog/quantization/>

**=====================================================================================**

**Que11:** Collinearity vs correlation?   
**Ans**: (<https://blog.clairvoyantsoft.com/correlation-and-collinearity-how-they-can-make-or-break-a-model-9135fbe6936a>)

Collinearity is a linear association between two predictors.

*Correlation is a statistical measure that indicates the extent to which two or more variables move together*. A positive correlation indicates that the variables increase or decrease together. A negative correlation indicates that if one variable increases, the other decreases, and vice versa.

*Covariance is another measure that describes the degree to which two variables tend to deviate from their means in similar ways****.***

**=====================================================================================**

**Que12**: What is pooling? Why do we use it? What are the types? (best explained in below link) (<https://towardsdatascience.com/convolutional-neural-network-17fb77e76c05>)

**=====================================================================================**

**Later answer the questions below by googling:**

KNN algorithm (<https://www.analyticsvidhya.com/blog/2021/05/20-questions-to-test-your-skills-on-k-nearest-neighbour/>)

Pooling downsamples the image in its height and width but the number of channels(depth) stays the same.

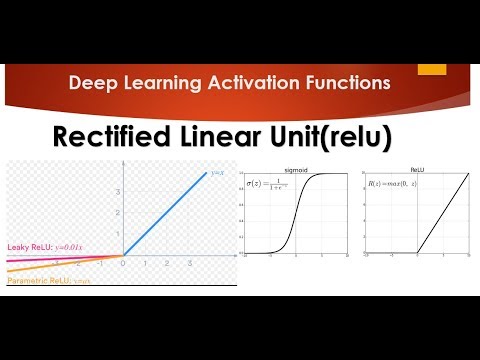
What are the different cost functions used for classification and regression problems? (<https://machinelearningknowledge.ai/cost-functions-in-machine-learning/>)

Why leaky relu introduced (dying neuron problem overcome), why relu over sigmoid(to solve vanishing gradient problem)?

### What are machine learning algorithms available in opencv ?

What is Vanishing and exploding gradient and how can it be overcome? [Tutorial 10- Activation Functions Rectified Linear Unit(relu) and Leaky Relu Part 2](https://www.youtube.com/watch?v=DDBk3ZFNtJc&t=532s)

Answer bit in below video



What is Multicollinearity? Ans best explained: (<https://www.analyticsvidhya.com/blog/2021/03/multicollinearity-in-data-science/>)

What are autoencoders? Their usage and real time applications of autoencoders?

What is GAN? Explain components of GAN?

What are transformers? (Question in gs labs)

What is an imbalanced dataset? How this situation can be overcome (ans: one of the solutions is do data augmentation, collect samples of the categories which has fewer samples in the dataset)

What is batch normalization?

What are performance metrics for classification and regression?

What is precision-recall curve? What is the ROC curve? What is AUC? What are type I and type II errors?

What are different optimization techniques? Explain SGD, Adam optimizer, adagrad

#### What's the difference between boosting and bagging and bootstrapping or what are the ensembling techniques?

What are different activation functions and why ReLU is used over sigmoid explain, how relu solves vanishing gradient problem explain in brief and also how dead neuron problem is solved by using leaky relu over relu activation function?

What is P-value and hypothesis?

Difference between ANN, CNN and RNN?

### ***30. Which algorithm is better at handling outliers' logistic regression or SVM?***

Logistic regression will find a linear boundary if it exists to accommodate the outliers. Logistic regression will shift the linear boundary in order to accommodate the outliers. SVM is insensitive to individual samples. There will not be a major shift in the linear boundary to accommodate an outlier. SVM comes with inbuilt complexity controls, which take care of overfitting. This is not true in case of logistic regression, so SVM is better in terms of handling the outliers and thus robust to outliers to make classification decisions.

### ***1. What is a logistic function? What is the range of values of a logistic function?***

f(z) = 1/(1+e -z )

The values of a logistic function will range from 0 to 1. The values of Z will vary from -infinity to +infinity.

### ***31. How will you deal with the multiclass classification problem using logistic regression?***

The most famous method of dealing with multiclass classification using logistic regression is using the one-vs-all approach. Under this approach, a number of models are trained, which is equal to the number of classes. The models work in a specific way. For example, the first model classifies the datapoint depending on whether it belongs to class 1 or some other class; the second model classifies the datapoint into class 2 or some other class. This way, each data point can be checked over all the classes.

Q. difference between YOLO and SSD? Explain RCNN, Fast RCNN and faster RCNN and what is single stage and 2 stage detectors (<https://www.analyticsvidhya.com/blog/2018/10/a-step-by-step-introduction-to-the-basic-object-detection-algorithms-part-1/>)

**### PROJECTS ###**

1. **Lattice ECP5 human detection:**

ECP5 is an FPGA Developed by Lattice Semiconductor

Performance metrics used is mAP and how it is used.

(Ref: <https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52>)

Dataset used: OIDV4 Google image OID dataset, downloaded using OIDv4 Tool Kit.

Model:

Squeezedet: <https://github.com/BichenWuUCB/squeezeDet>

* SqueezeDet is an object detection model that was introduced in 2017. It is designed to perform real-time object detection on resource-constrained devices, such as embedded systems and autonomous vehicles, where computational power and memory are limited.
* The main idea behind SqueezeDet is to reduce the computational complexity of the object detection pipeline while maintaining competitive accuracy. It achieves this by utilizing a lightweight convolutional neural network architecture inspired by the SqueezeNet model, which is known for its small size and efficient computation.
* Here are the key components and characteristics of the SqueezeDet model:

1. **SqueezeNet-Based Architecture**: SqueezeDet utilizes a modified version of the SqueezeNet architecture as its backbone. SqueezeNet replaces the standard 3x3 filters in convolutional layers with 1x1 filters to reduce the number of parameters. This significantly reduces the model's size and computational requirements.
2. **Fire Modules**: SqueezeDet introduces "Fire Modules" into the SqueezeNet architecture. A Fire Module consists of a squeeze layer, which performs 1x1 convolutions to reduce the number of input channels, followed by expand layers that consist of 1x1 and 3x3 convolutions. The Fire Modules help capture and propagate spatial information effectively while maintaining model compactness.
3. **Detection Head**: SqueezeDet appends a detection head to the SqueezeNet backbone. The detection head performs object detection by predicting bounding boxes and class probabilities. It consists of a set of convolutional and fully connected layers that process the feature maps from the backbone network to generate the final detection outputs.
4. **Anchor Boxes**: SqueezeDet uses anchor boxes, which are predefined bounding boxes of different scales and aspect ratios, to improve localization accuracy. These anchor boxes serve as reference templates during training and inference. The model predicts the offsets and confidences for each anchor box to determine the final bounding box predictions.
5. **Loss Function**: SqueezeDet uses a combination of classification loss and bounding box regression loss to train the model. The classification loss measures the difference between predicted class probabilities and ground truth labels. The bounding box regression loss measures the discrepancy between predicted bounding box coordinates and ground truth boxes.

* SqueezeDet demonstrates real-time object detection capabilities with reasonable accuracy while maintaining a small model size. It has been used in various applications, particularly in autonomous driving and robotics, where computational efficiency is crucial.

Squeezenet (<https://towardsdatascience.com/review-squeezenet-image-classification-e7414825581a>)

Framework: Tensorflow

Tools: SensAI2.0 for compiling, analyzing, simulating the model and for firmware generation and diamond programmer tool by Lattice for flashing hardware bit file to ECP5 and deploying the model.

Participation: Exploring the image dataset and also data annotation tool (LabelImg tool), Documentation, checking the PC inference on test images and supervising the training and finetuning.

1. **Object detection using YOLO v3:**

Refer:

<https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b> (yolo introduction)

<https://towardsdatascience.com/yolo-v3-object-detection-with-keras-461d2cfccef6> (For code of object detection with YOLO v3)

* Explain the history of YOLO (v1, v2 and v3). Also check RCNN, Faster RCNN and SSD.
* Explain the feature changes in YOLO v3 from YOLO v2 as explained below:

1. Different backbone architecture (I.e Darknet 53 + 53 object detection layers = 106, in YOLOv2 its Darknet 19 with 11 object detection layers). Thus, performing slower than v2.

2. Object detections at 3 different scales with detection kernel with shape of detection kernel 1x1x(Bx(5+C))

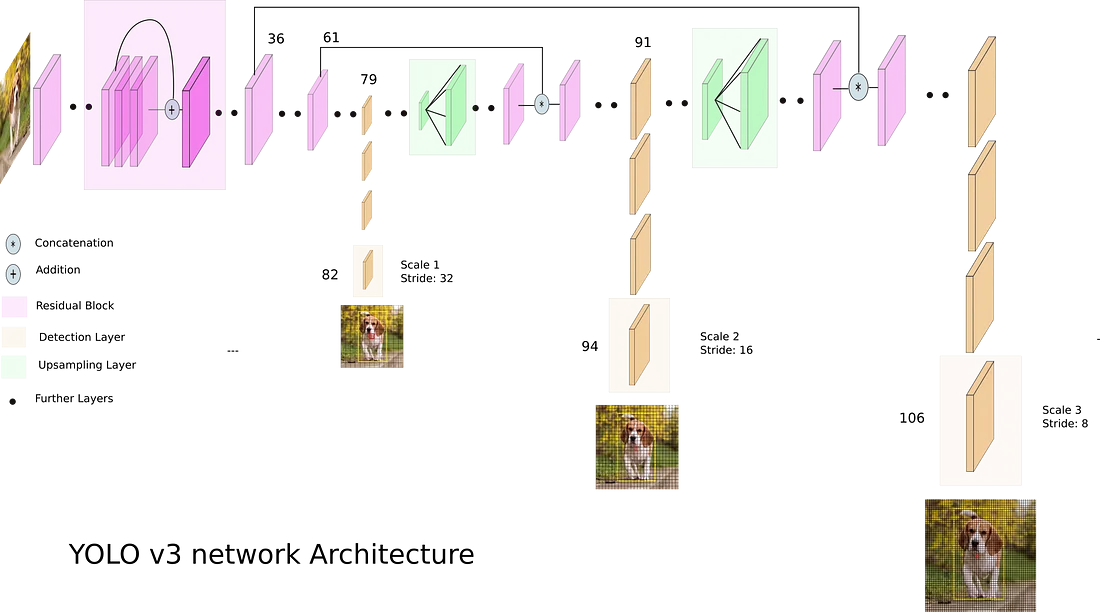
3. Better at detecting smaller objects (resolves the frequent complaints from YOLO v2, In YOLO v3 the upsampled layers are concatenated with the previous layers to preserve the fine-grained features)

4. Choice of Anchor boxes (YOLO v3 uses 9 anchor boxes, 3 at each scale).

5. More Bounding Boxes per image.

6. Change in Loss functions (uses cross entropy instead of squaring errors, object score and class probabilities are now measured with logistic regression)

7. No more softmaxing the classes (Now, YOLO v3 uses multilabel classification. The authors of YOLO have refrained from softmaxing the classes. Instead, each class score is predicted using logistic regression and a threshold is used to predict multiple labels for an object. Classes with scores higher than this threshold are assigned to the box.)



1. **Text detection using Mobilenet SSD:**

Learn more about MobileNet model and SSD detector

<https://softnautics-my.sharepoint.com/:p:/r/personal/ashwini_porkute_softnautics_com/_layouts/15/Doc.aspx?sourcedoc=%7BC07B9632-0295-4767-B863-DB7DDDA530E0%7D&file=TextDetection_11Sept.pptx&action=edit&mobileredirect=true&cid=327d6ea8-8de3-4336-a08c-06115a424684>

1. **Optimization and implementation of Vision kernels for ZSP:**

**OpenVX:** it is an open, royalty-free standard for cross platform acceleration of computer vision applications. It uses a connected graph representation of operations.

OpenVX is based on a connected graph of vision nodes that can execute the preferred chain of operations.

**OpenVX Conformance Test Suite (CTS)**: The purpose of the CTS is to validate that an OpenVX implementation meets the required functionality and behavior as defined by the standard.

Eclipse IDE is used for programming in C.

There were 2 teams (ZSP and Host), I was in ZSP team which works on implementation of vision kernels and testing the same with binary image. Whereas the Host team checks and validates the same with the CTS framework.

In ZSP side, we redefined the Vision kernels as per our code skeleton and after validation of these kernels from Host side, also we created the static library (libzspapi.a) where all validated kernel object files were linked and created the executable.

All team members were allocated with specific kernels to implement and optimize (to lower the ZSP cycles).

I was allocated with the following kernels:

* Convolution
* AND, OR, NOT, XOR
* ConvertBitDepth, AbsDiff
* TensorMatrixMultiply, TensorMultiply, TensorAdd, TensorSubtract, TensorLUT

**Memory Access Optimization**: DSP platforms typically have specific memory hierarchies and access patterns. Optimizing memory access patterns, utilizing local memory, and minimizing data transfers between different memory types (e.g., RAM, cache, or on-chip memory) can significantly improve performance on DSP platforms.

**Tiling and Loop Unrolling:** DSP platforms often benefit from tiling and loop unrolling techniques to optimize memory access and minimize loop overhead. Breaking down the computations into smaller tiles and unrolling loops can enable efficient vectorization and improve data reuse, leading to performance gains.

**Compiler Optimizations**: Take advantage of the compiler optimizations available for the ZSP architecture. Enable optimizations such as loop unrolling, instruction scheduling, and automatic vectorization

<https://softnautics-my.sharepoint.com/:w:/p/ashwini_porkute/EStq52_szK5BmyCSSQ7JFocB8fOj9s4x24tUFMfOacpVZg?e=RaG80t>

<https://softnautics-my.sharepoint.com/:w:/p/ashwini_porkute/EYKcCLZbO8FFkuSfxi4FsAIBqPtNaWH_nPgEj1m8zWfAQw?e=Ro2gzP>

PYTHON CODING INTERVIEW QUESTIONS AND ANSWERS:

Link : <https://codingcompiler.com/python-coding-interview-questions-answers/>

Blur or smoothen the image using box filter, gaussian filter, median, or LPF (low pass filter)

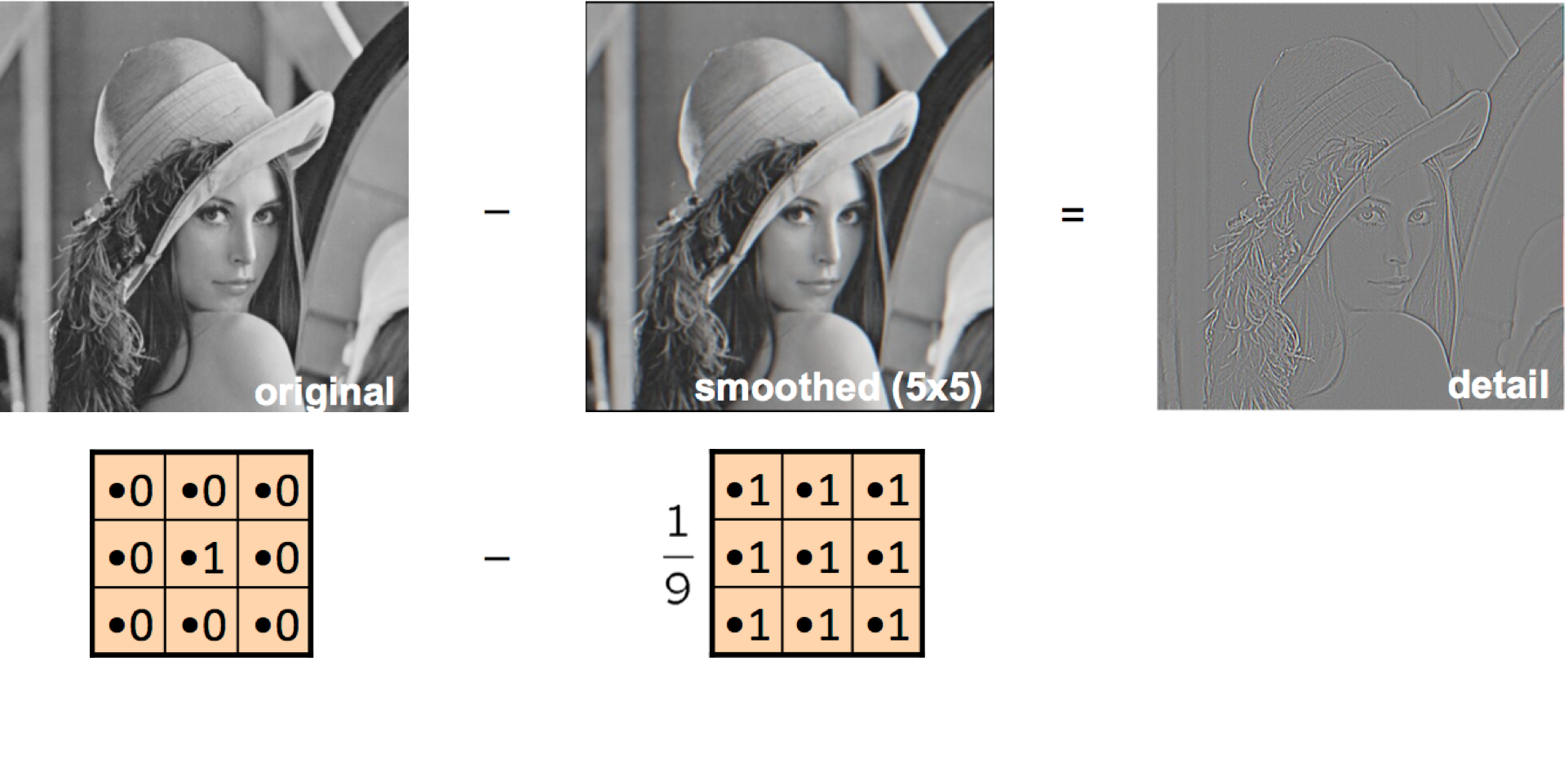
Sharpening the image is nothing but using HPF (high pass filter)

Box filter = to blur the image

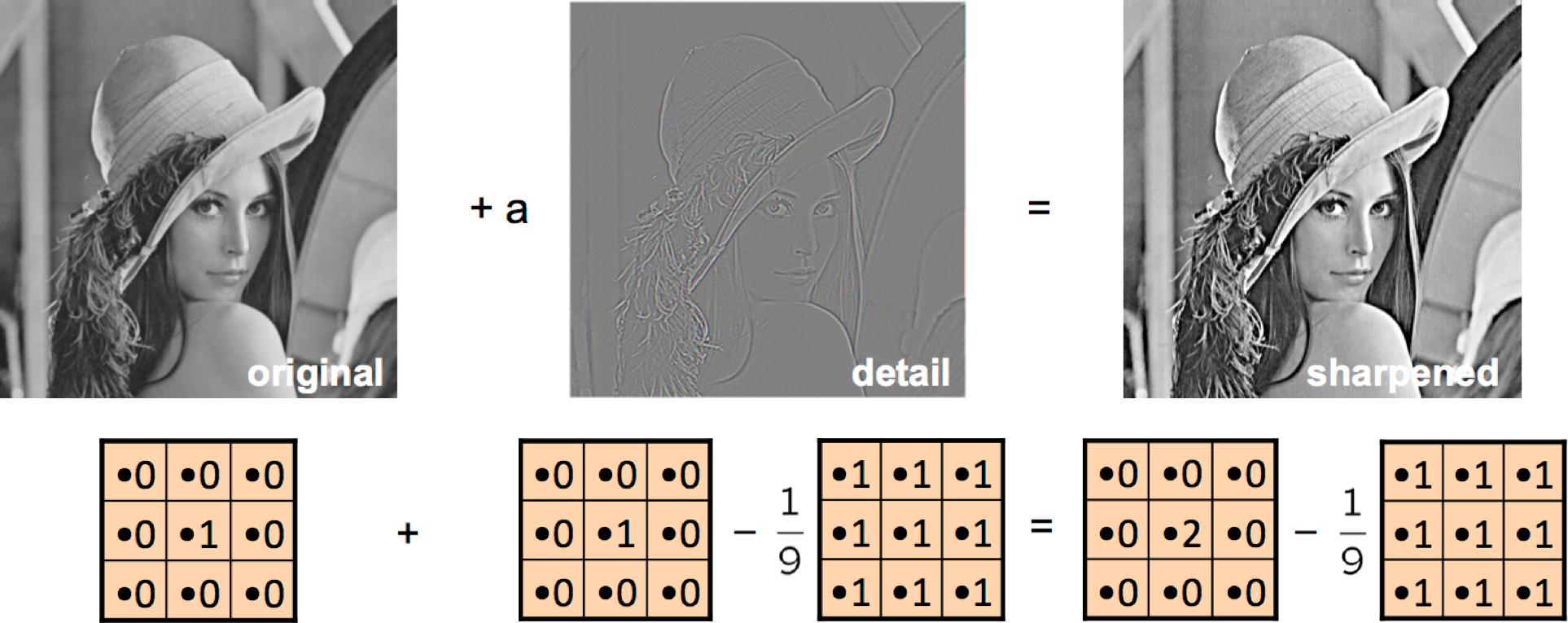
Sharpening filter:

A sharpening filter can be broken down into two steps: It takes a smoothed image, subtracts it from the original image to obtain the "details" of the image, and adds the "details" to the original image.

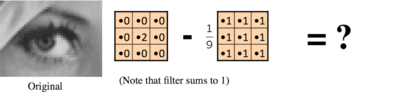
Step 1: Original - Smoothed = "Details"



Step 2: Original + "Details" = Sharpened



Result:



Canny edge detection steps:

1. It is a multi-stage algorithm and we will go through each stage.
2. **Noise Reduction:** Since edge detection is susceptible to noise in the image, first step is to remove the noise in the image with a 5x5 Gaussian filter. We have already seen this in previous chapters.
3. **Finding Intensity Gradient of the Image**

Smoothened image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction (Gx) and vertical direction (Gy). From these two images, we can find edge gradient and direction for each pixel as follows:

*Edge*\_*Gradient*(*G*)=*G*2*x*+*G*2*y*−−−−−−−√*Angle*(*θ*)=tan−1(*GyGx*)

Gradient direction is always perpendicular to edges. It is rounded to one of four angles representing vertical, horizontal and two diagonal directions.

1. **Non-maximum Suppression**

After getting gradient magnitude and direction, a full scan of image is done to remove any unwanted pixels which may not constitute the edge. For this, at every pixel, pixel is checked if it is a local maximum in its neighborhood in the direction of gradient. Check the image below:

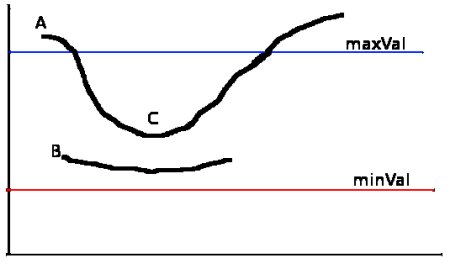


Point A is on the edge (in vertical direction). Gradient direction is normal to the edge. Point B and C are in gradient directions. So, point A is checked with point B and C to see if it forms a local maximum. If so, it is considered for next stage, otherwise, it is suppressed (put to zero).

In short, the result you get is a binary image with "thin edges".

1. **Hysteresis Thresholding**

This stage decides which are all edges are really edges and which are not. For this, we need two threshold values, minVal and maxVal. Any edges with intensity gradient more than maxVal are sure to be edges and those below minVal are sure to be non-edges, so discarded. Those who lie between these two thresholds are classified edges or non-edges based on their connectivity. If they are connected to "sure-edge" pixels, they are considered to be part of edges. Otherwise, they are also discarded. See the image below:

1. 

The edge A is above the maxVal, so considered as "sure-edge". Although edge C is below maxVal, it is connected to edge A, so that also considered as valid edge and we get that full curve. But edge B, although it is above minVal and is in same region as that of edge C, it is not connected to any "sure-edge", so that is discarded. So it is very important that we have to select minVal and maxVal accordingly to get the correct result.

This stage also removes small pixels noises on the assumption that edges are long lines.

Erosion, dilation, opening, closing, morphological gradient:

<https://docs.opencv.org/master/d9/d61/tutorial_py_morphological_ops.html>

Histogram, and histogram equalization:

<https://towardsdatascience.com/histogram-equalization-5d1013626e64>