AI/ML BOOTCAMP

William Phelps

Christopher Newport University/Jefferson Lab

CHRISTOPHER NEWPORT UNIVERSITY

Jefferson Lab
Thomas Jefferson National Accelerator Facility

Course Topics*

1 Semester DS/AI Course in 4 lectures
45 hours in 8 hours

- **Lectures 1 & 2**
  - **Lecture 1**
    - Introduction to workflow and tools (git/anaconda/pycharm)
    - Introduction to Data Science
    - Python Review with jupyter notebooks
  - **Lecture 2**
    - Data Visualization
    - Principles of data visualization from Edward Tufte
    - Pandas introduction
- **Lectures 3 & 4**
  - **Lecture 3**
    - Machine Learning Introduction
    - Regression/Curvefit
    - Overfitting/Underfitting
    - kNN classifier
    - MNIST/Handwritten digit classifier
  - **Lecture 4**
    - Overfitting/Underfitting
    - Early Stopping
    - Hyperparameter optimization
    - Convolutional Neural Networks
    - CIFAR 10 datasets

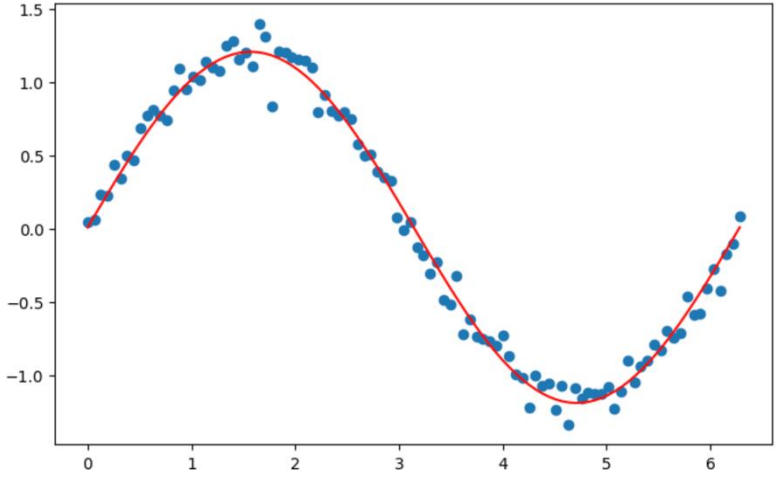
*Tentative

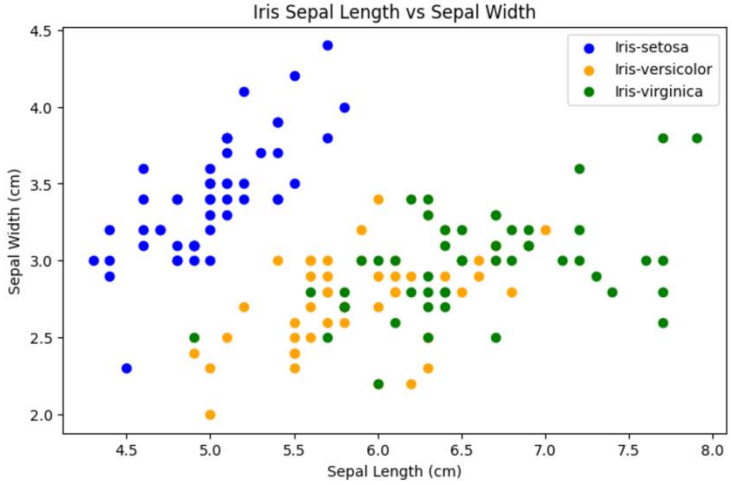
RECAP/MNIST

Recap – Regression and kNN

```
# Let's fit with curve fit and extract parameters
from scipy.optimize import curve_fit
def func(x, a, b):
    return a*np.sin(x) + b
popt, pcov = curve_fit(func, x, y)
```

```
# Plot function on top of dataset
plt.plot(x, y, 'o')
plt.plot(x, func(x, *popt), 'r-');
```





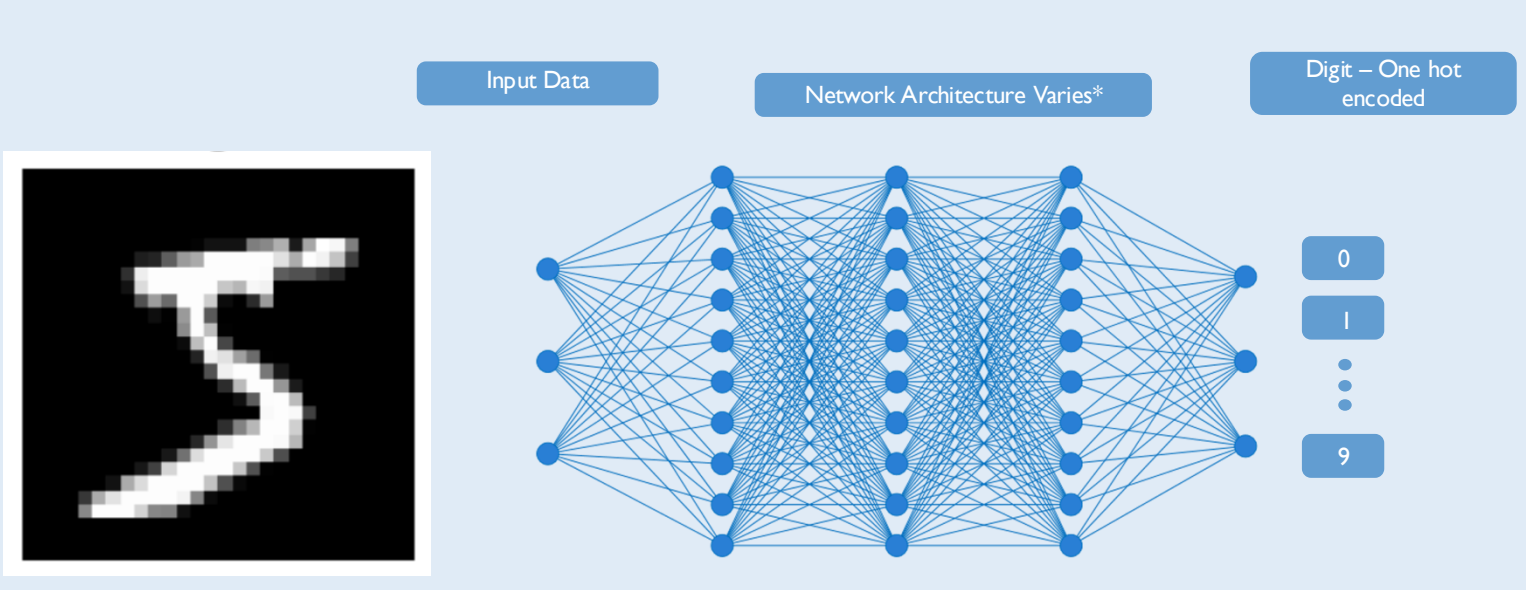

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
```

Recap – MNIST Dataset

- MNIST handwritten digit database
- (Modified National Institute of Standards and Technology dataset)
- 60,000 handwritten digits from the census bureau for training
- 10,000 test images
- Each image is 28x28 pixels and has a corresponding label

MLP Example – MNIST

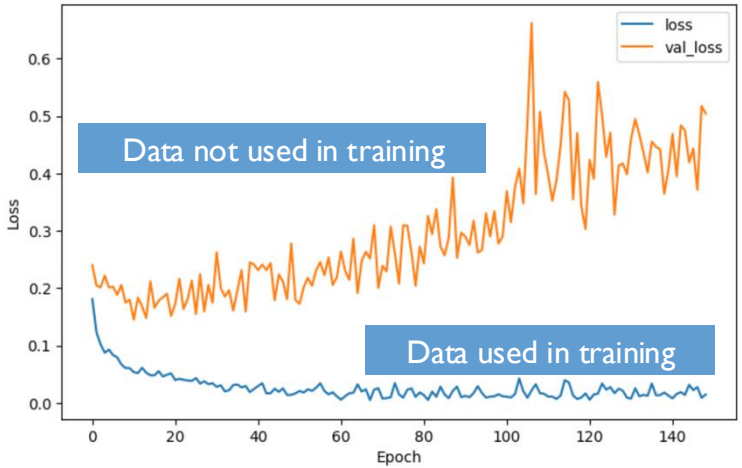
- Read in 28x28 images of handwritten digits
- Model Architecture: MLP, relu activation function



MNIST MLP

Is this the best we can do with a MLP?

```
# Build a MLP
model = keras.Sequential()
model.add(layers.Flatten(input_shape=(28, 28)))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
model.summary();

 # Compile model
 model.compile(optimizer='adam',
               loss='sparse_categorical_crossentropy',
               metrics=['accuracy'])

 # Train the model
 history = model.fit(train_images,
                     train_labels,
                     epochs=150,
                     batch_size=128,
                     validation_split=0.2)
```



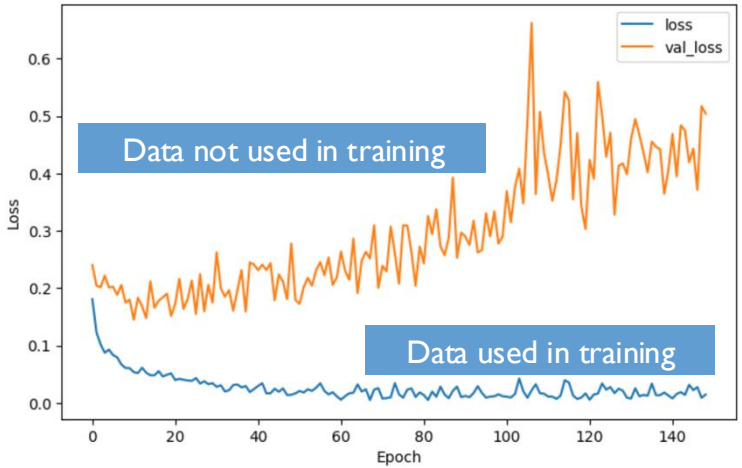
How many epochs?

How well did we do?

MNIST MLP

Is this the best we can do with a MLP?

```
# Build a MLP
model = keras.Sequential()
model.add(layers.Flatten(input_shape=(28, 28)))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
model.summary();

 # Compile model
 model.compile(optimizer='adam',
               loss='sparse_categorical_crossentropy',
               metrics=['accuracy'])

 # Train the model
 history = model.fit(train_images,
                     train_labels,
                     epochs=150,
                     batch_size=128,
                     validation_split=0.2)
```



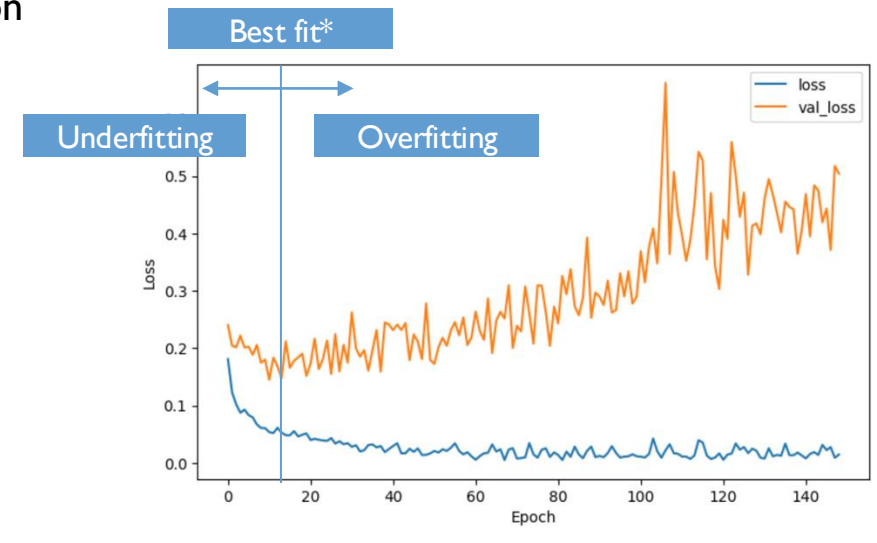
How many epochs?

How well did we do?

Early Stopping with Callbacks

- Callbacks monitor training and trigger actions during model.fit().
- Early stopping stops training when a chosen metric stops improving.
- Here, monitor='val_loss' tracks validation loss.
- patience=3 allows 3 epochs without improvement before stopping.
- This helps reduce overfitting and saves training time.

```
# Example of early stopping with callbacks
from tensorflow.keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(monitor='val_loss', patience=3)
history = model.fit(train_images,
                    train_labels,
                    epochs=50,
                    validation_split=0.2,
                    callbacks=[early_stopping])
```

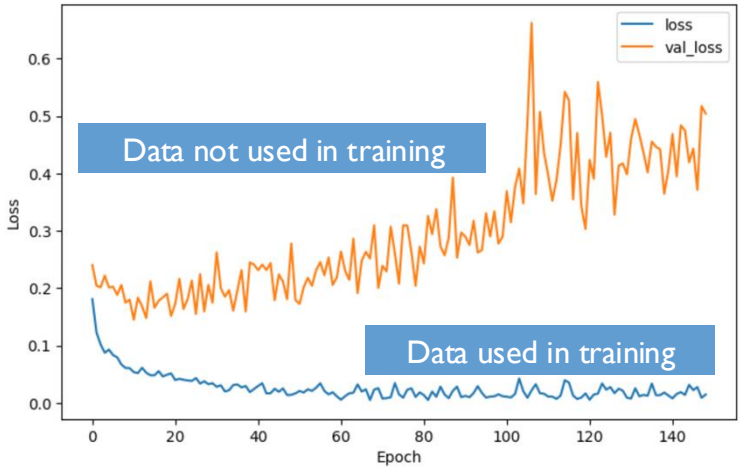


* Somewhat debatable in this example

MNIST MLP

Is this the best we can do with a MLP?

```
# Build a MLP
model = keras.Sequential()
model.add(layers.Flatten(input_shape=(28, 28)))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
model.summary();

 # Compile model
 model.compile(optimizer='adam',
               loss='sparse_categorical_crossentropy',
               metrics=['accuracy'])

 # Train the model
 history = model.fit(train_images,
                     train_labels,
                     epochs=150,
                     batch_size=128,
                     validation_split=0.2)
```



How many epochs?

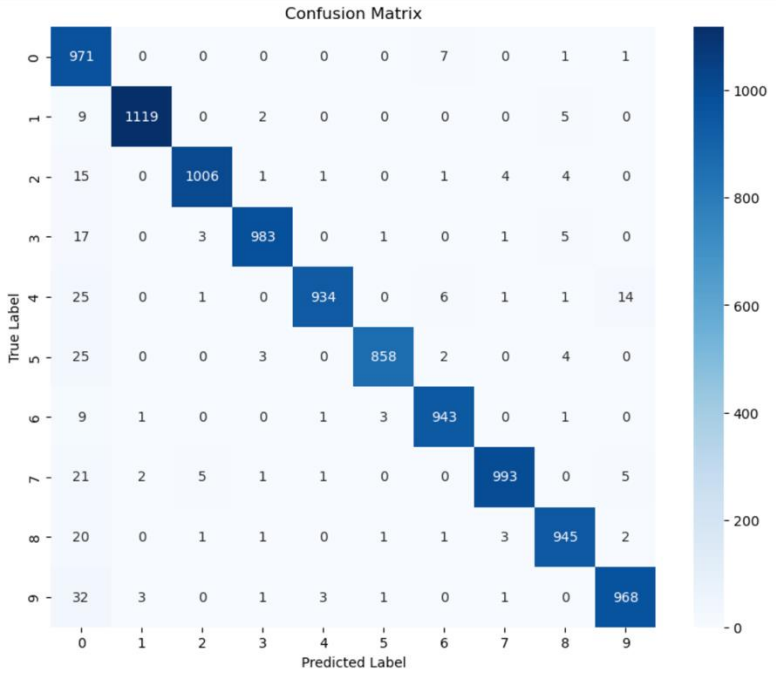
How well did we do?

Accuracy and Confusion Matrix

- model.evaluate(test_images, test_labels) measures performance on unseen test data.
- It returns the test loss and any metrics specified during model.compile(), such as accuracy.
- The test data should not be used in training!
- Next we will look at the confusion matrix

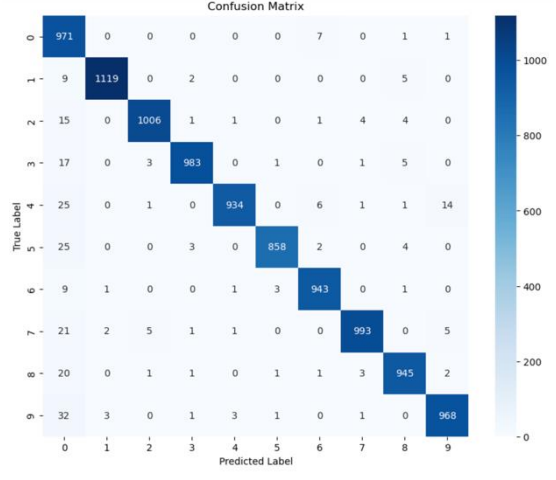
```
# Calculate accuracy on the test set
test_loss, test_acc = model.evaluate(test_images, test_labels)
```

```
313/313 [==============================] - 1s 4ms/step - loss: 0.3390 - accuracy: 0.9720
```



Confusion Matrix Cont'd

- model.predict(test_images) generates class probabilities for each test image.
- argmax(axis=1) converts probabilities into predicted class labels.
- The confusion matrix compares true vs. predicted labels
- strong performance appears as large values along the diagonal.



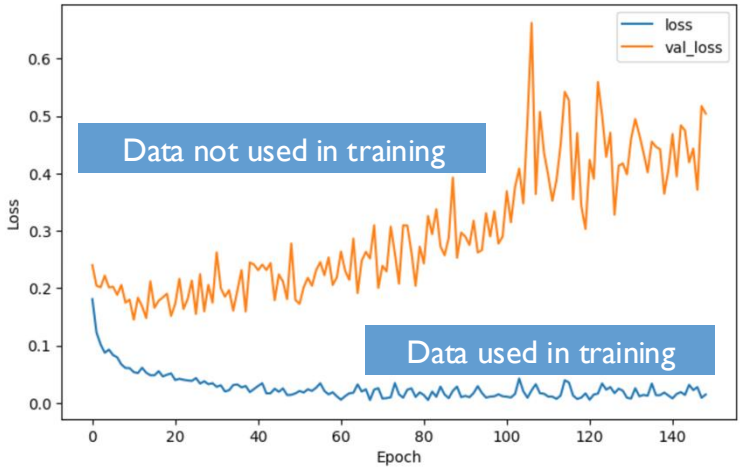

```
# Plot confusion matrix
from sklearn.metrics import confusion_matrix
import seaborn as sns
predictions = model.predict(test_images)
predicted_labels = predictions.argmax(axis=1)
cm = confusion_matrix(test_labels, predicted_labels)
plt.figure(figsize=(10, 8))
sns.heatmap(cm,
            annot=True,
            fmt='d',
            cmap='Blues',
            xticklabels=range(10),
            yticklabels=range(10))
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show();
```

MNIST MLP

Is this the best we can do with a MLP?

```
# Build a MLP
model = keras.Sequential()
model.add(layers.Flatten(input_shape=(28, 28)))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
model.summary();
```

```
# Compile model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```




```
# Train the model
history = model.fit(train_images,
                    train_labels,
                    epochs=150,
                    batch_size=128,
                    validation_split=0.2)
```

How many epochs?

How well did we do?

What is a hyperparameter?

- They are not learned directly from the data.
- They control the model architecture or training process.
- **Examples:** number of layers, number of neurons, learning rate, batch size, dropout rate, optimizer, and number of epochs.
- **Hyperparameter optimization** searches for settings that improve validation performance.
- In contrast, **model parameters are learned during training**, such as weights and biases.

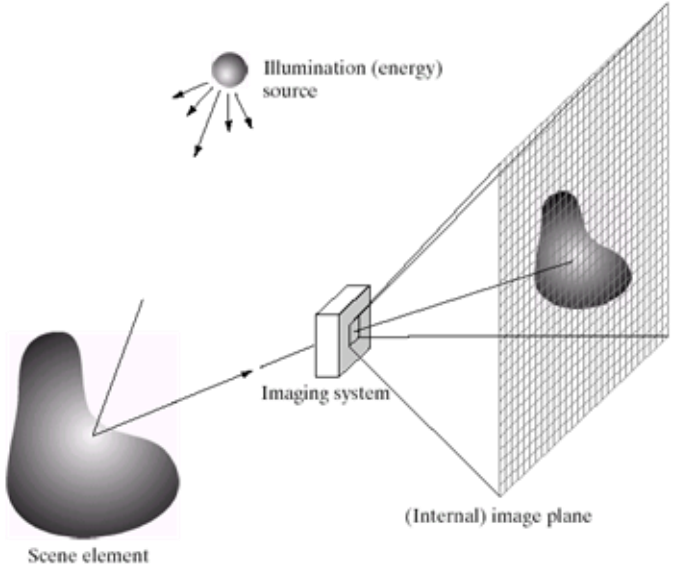
Example Hyperparameters

```
# Build a MLP
model = keras.Sequential()
model.add(layers.Flatten(input_shape=(28, 28)))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
model.summary();

# Compile model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
history = model.fit(train_images,
                    train_labels,
                    epochs=150,
                    batch_size=128,
                    validation_split=0.2)
```

Hyperparameter tuning Example

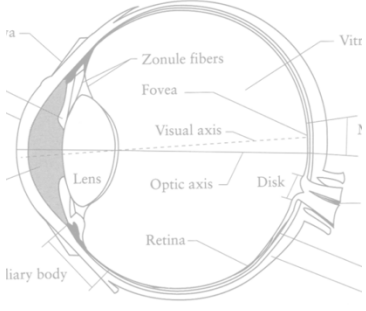
IMAGES AND CONVOLUTIONS

What is an image?



Digital Camera

https://en.wikipedia.org/wiki/Pointillism



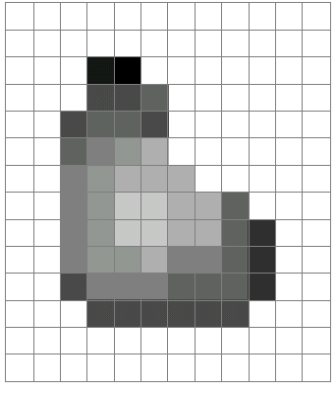
The Eye

Original slide by Noah Snavely (Cornell)

Source: A. Efros

What is a digital image?

- A grid (matrix) of intensity values



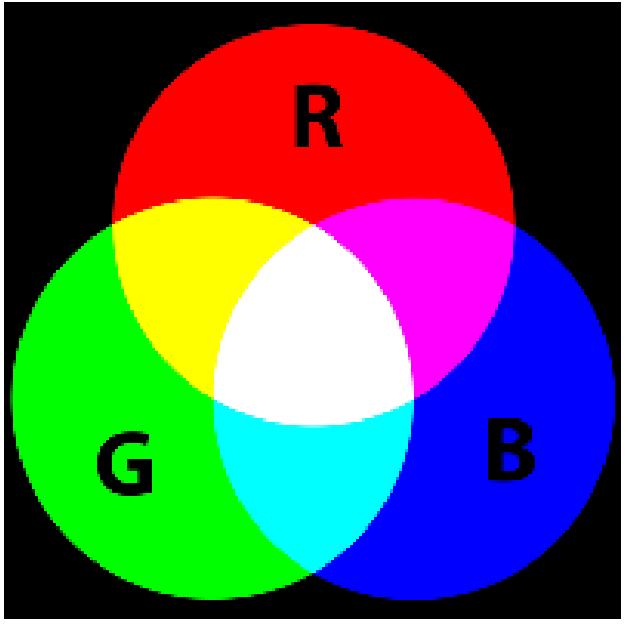
=

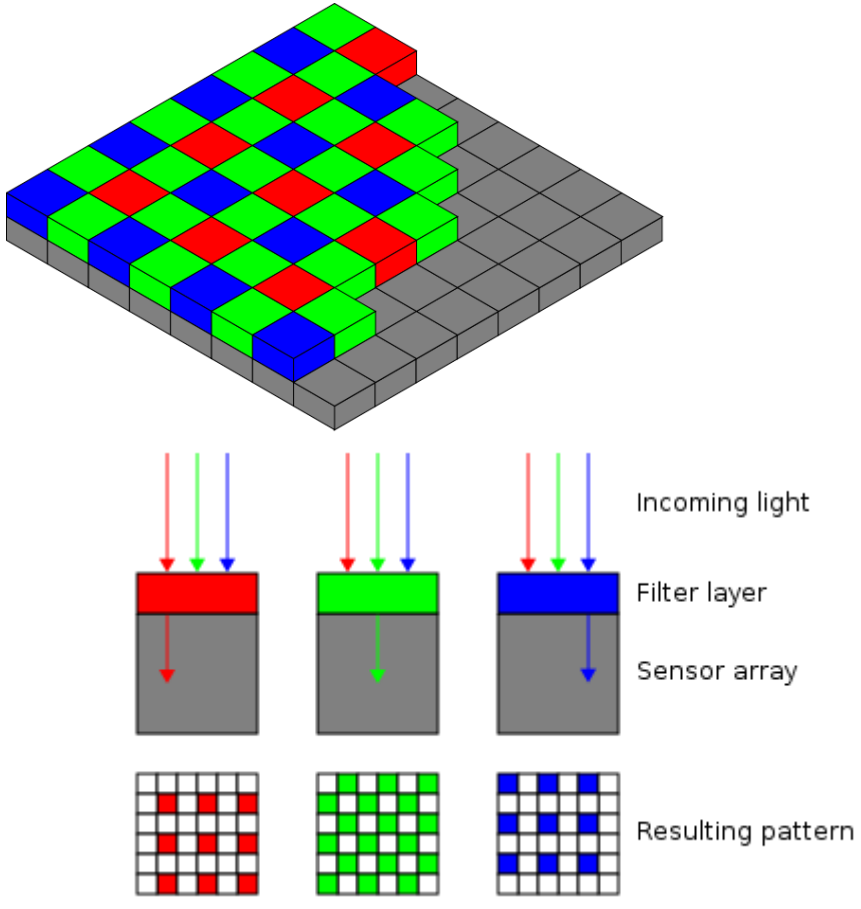
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	20	0	255	255	255	255	255	255	255
255	255	255	75	75	75	255	255	255	255	255	255
255	255	75	95	95	75	255	255	255	255	255	255
255	255	96	127	145	175	255	255	255	255	255	255
255	255	127	145	175	175	175	255	255	255	255	255
255	255	127	145	200	200	175	175	95	255	255	255
255	255	127	145	200	200	175	175	95	47	255	255
255	255	127	145	145	175	127	127	95	47	255	255
255	255	74	127	127	127	95	95	95	47	255	255
255	255	255	74	74	74	74	74	74	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255

(common to use one byte per value: 0 = black, 255 = white)

Original slide by Noah Snavely (Cornell)

Color Images





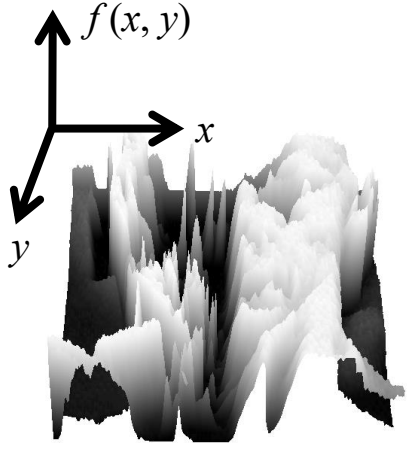
https://en.wikipedia.org/wiki/Bayer_filter

https://en.wikipedia.org/wiki/RGB_color_model#/media/File:Rgb-compose-Alim_Khan.jpg

What is an image?

- We can think of a (grayscale) image as a **function**, f, from $\mathbb{R}^2$ to $\mathbb{R}$:
  - $f(x,y)$ gives the **intensity** at position (x,y)

snoop



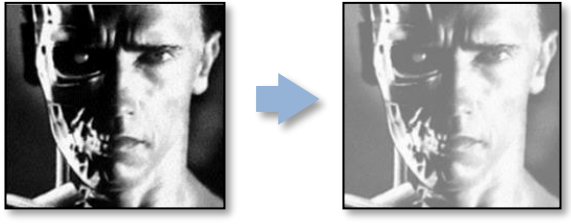
3D view

  - A **digital** image is a discrete (**sampled**, **quantized**) version of this function

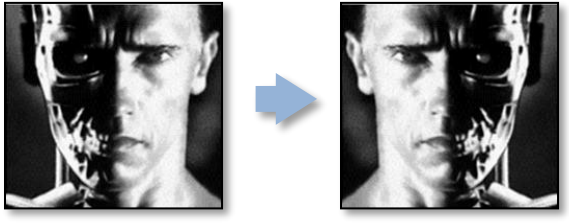
Original slide from Noah Snavely (Cornell)

Image transformations

- As with any function, we can apply operators to an image



$g(x,y) = f(x,y) + 20$



$g(x,y) = f(-x,y)$

- We'll talk about a special kind of operator, *convolution* (linear filtering)

Original slide from Noah Snavely (Cornell)

Image filtering

- Modify the pixels in an image based on some function of a local neighborhood of each pixel

Local image data

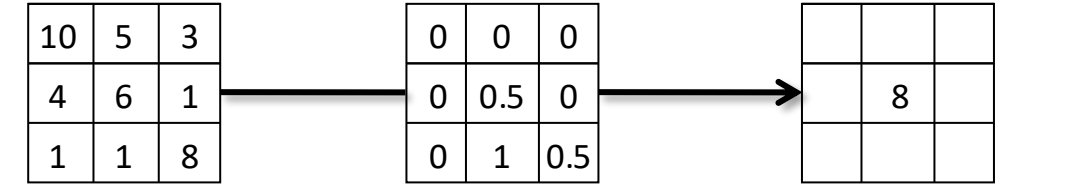
Modified image data

Original slide from Noah Snavely (Cornell)

Source: L. Zhang

Linear filtering

- One simple version: linear filtering (cross-correlation, convolution)
  - Replace each pixel by a linear combination (a weighted sum) of its neighbors
- The prescription for the linear combination is called the "kernel" (or "mask", "filter")



Original slide by Noah Snavely (Cornell)

Source: L. Zhang

Cross-correlation

Let F be the image, H be the kernel (of size 2*k*+1 x 2*k*+1), and G be the output image

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u,j+v]$$

This is called a **cross-correlation** operation:

$$G = H \otimes F$$

- Can think of as a "dot product" between local neighborhood and kernel for each pixel

Original slide by Noah Snavely (Cornell)

Convolution

- Same as cross-correlation, except that the kernel is "flipped" (horizontally and vertically)

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u,j-v]$$

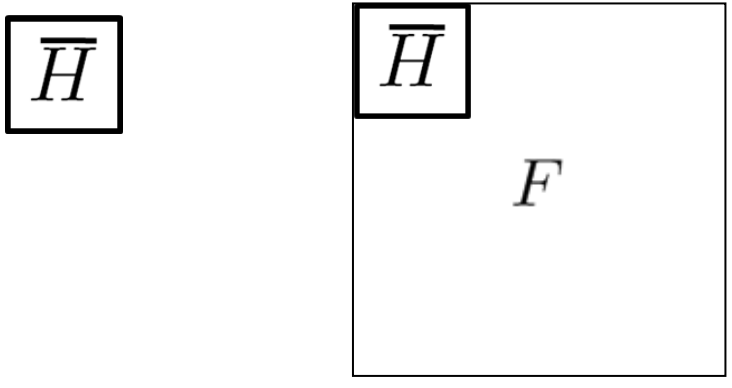
This is called a **convolution** operation:

$$G = H * F$$

- Convolution is **commutative** and **associative**
  - Can combine convolution operations

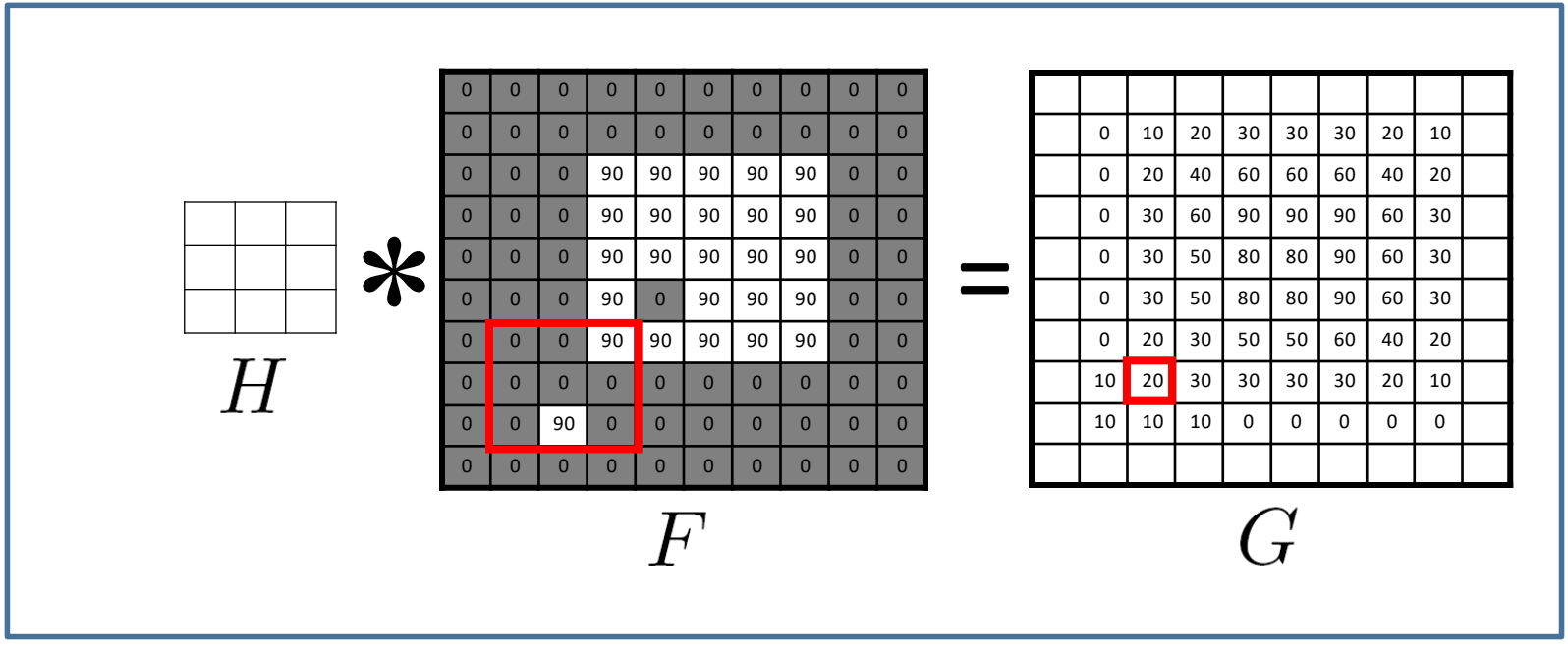
Original slide by Noah Snavely (Cornell)

Convolution



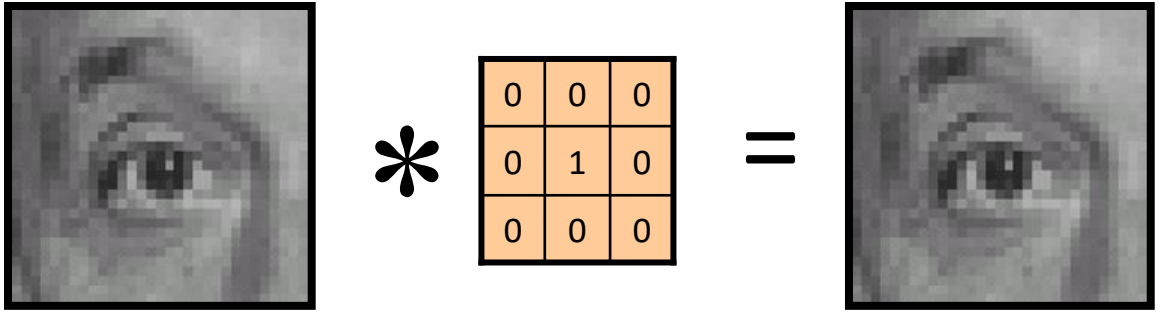
Original slide by Noah Snavely (Cornell) Adapted from F. Durand

Mean filtering



Original slide by Noah Snavely (Cornell)

Linear filters: examples



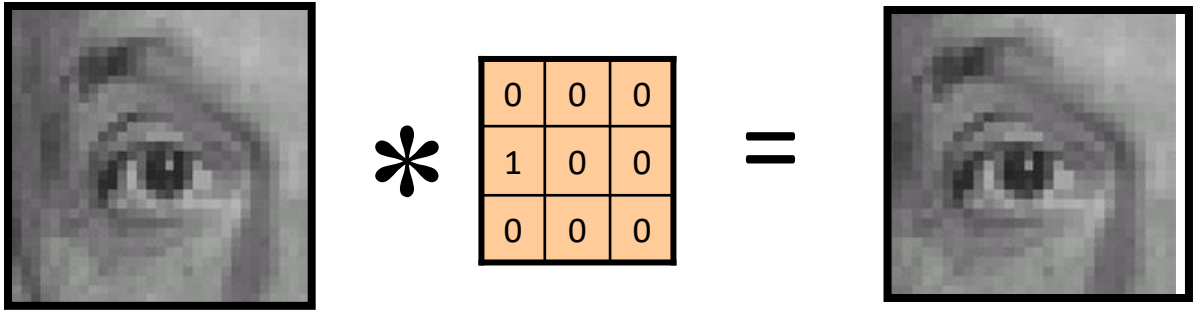
Original

Identical image

Original slide by Noah Snavely (Cornell)

Source: D. Lowe

Linear filters: examples



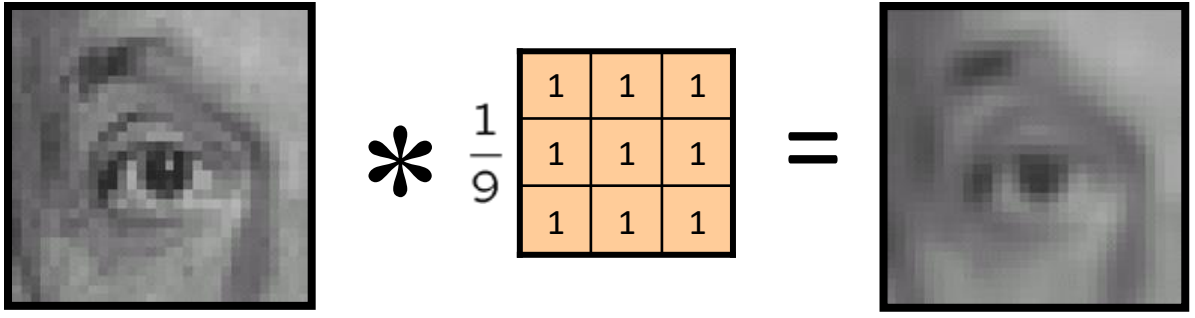
Original

Shifted left
By 1 pixel

Original slide by Noah Snavely (Cornell)

Source: D. Lowe

Linear filters: examples



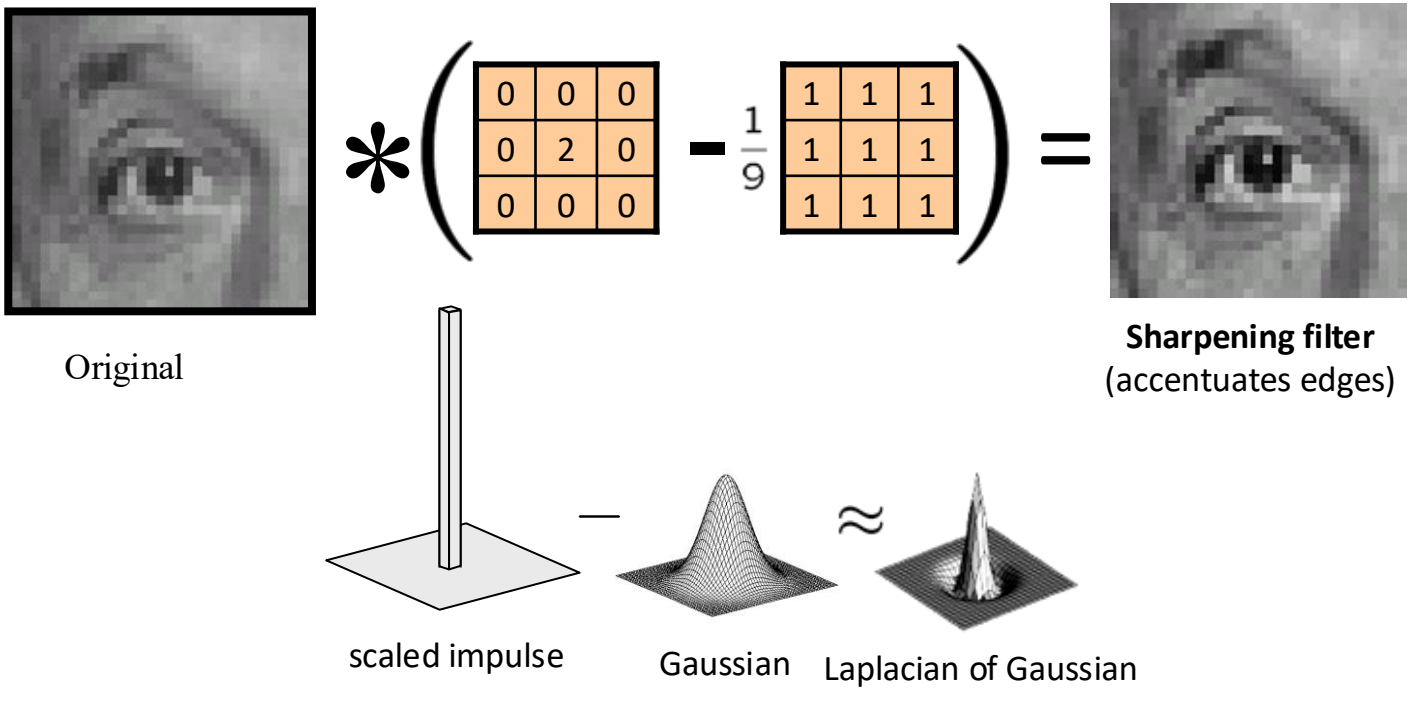
Original

Blur (with a mean filter)

Original slide by Noah Snavely (Cornell)

Source: D. Lowe

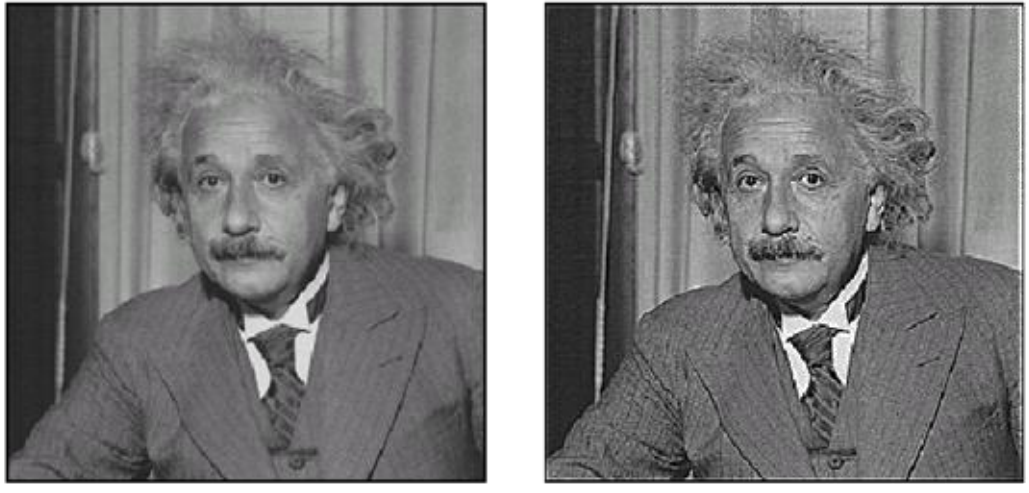
Linear filters: examples



Original slide by Noah Snavely (Cornell)

Source: D. Lowe

Sharpening

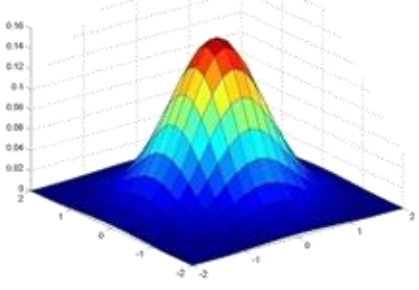


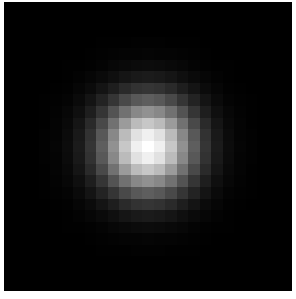
before after

Original slide by Noah Snavely (Cornell)

Source: D. Lowe

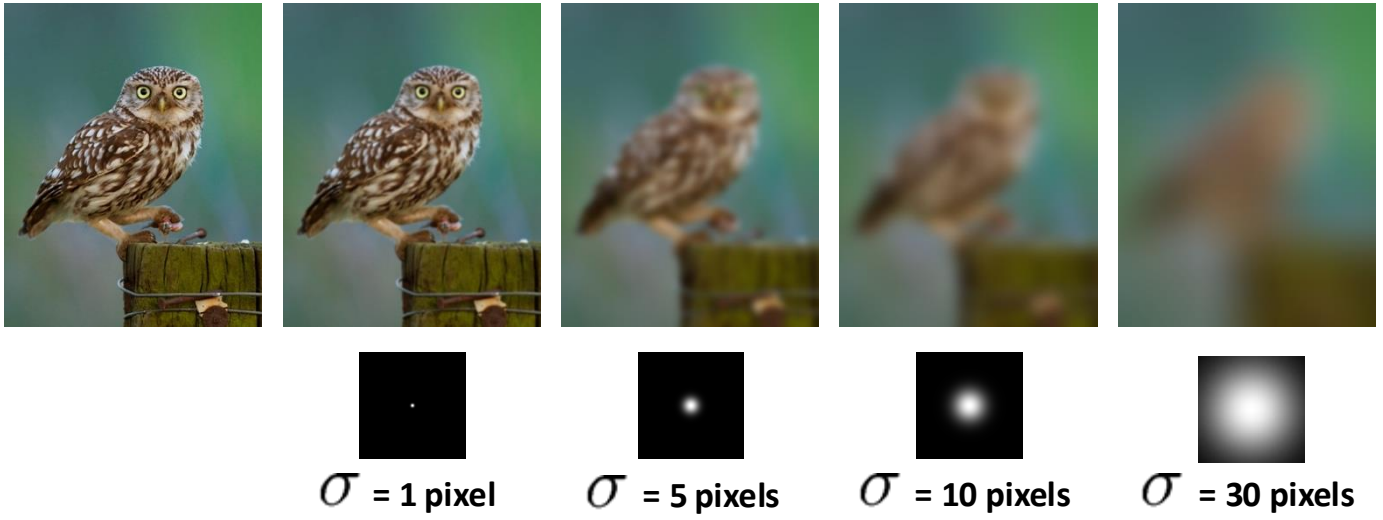
Gaussian Kernel





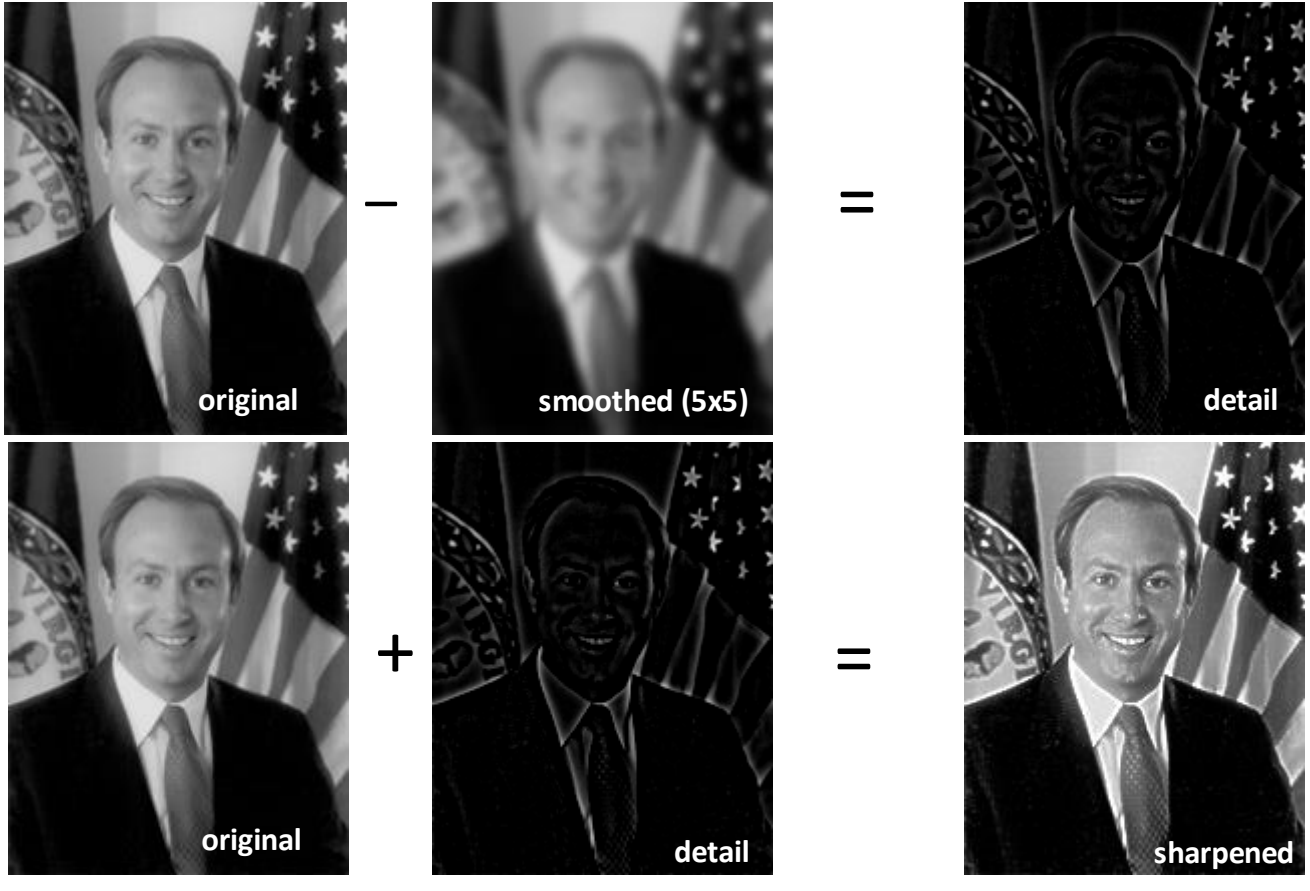
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Gaussian filters



Original slide by Noah Snavely (Cornell)

Sharpening revisited

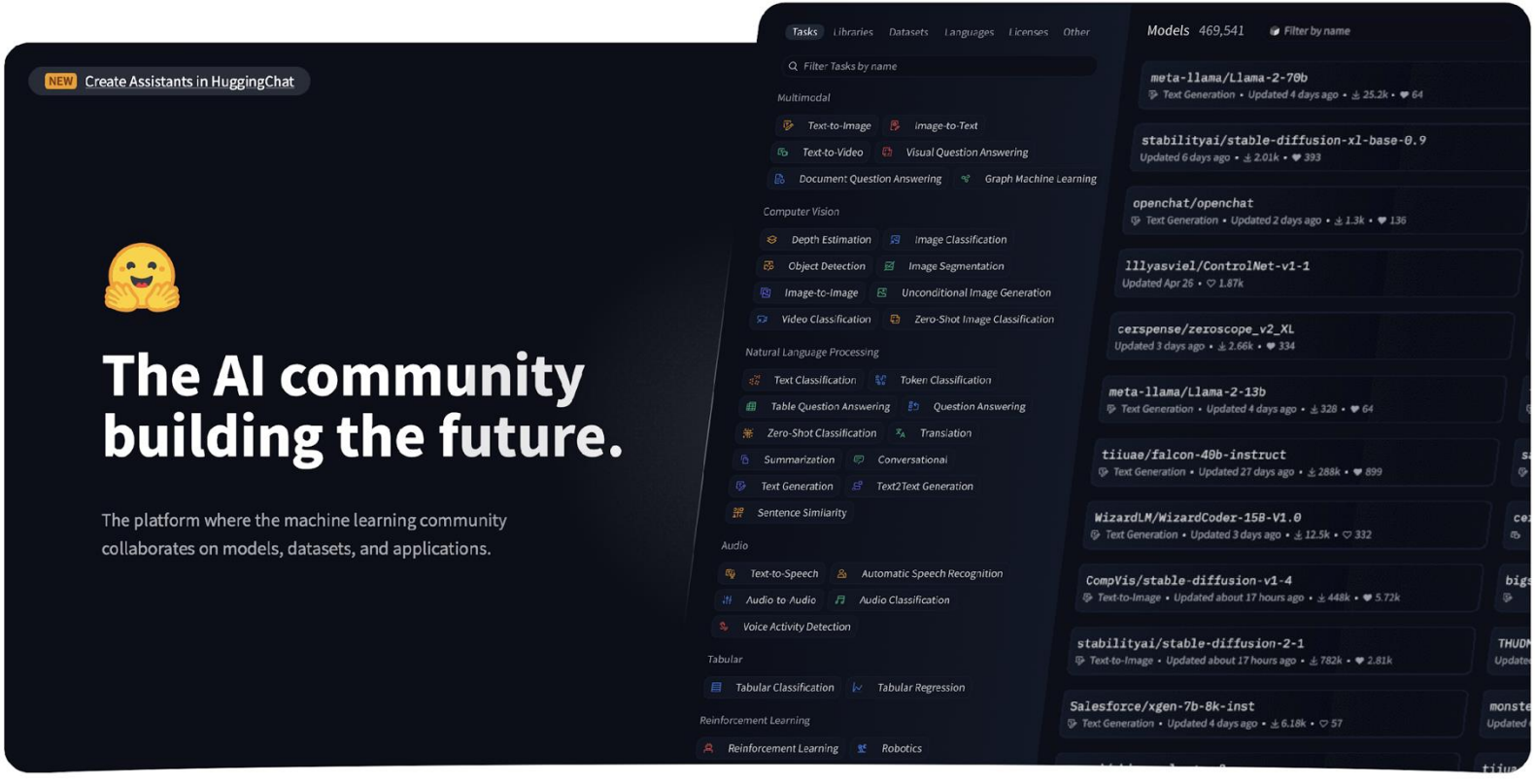


What does blurring take away?

Based on a slide from Noah Snavely (Cornell)

CNN Demo with MNIST

Proliferation of Pretrained Models

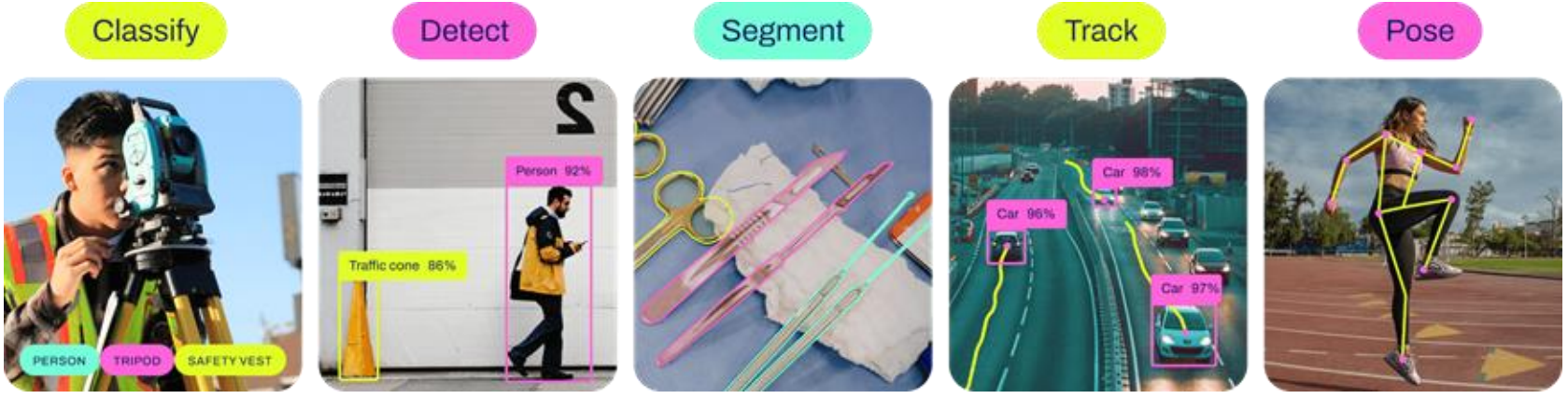


https://huggingface.co

YOLOv8

- YOLOv8 (You Only Look Once, version 8) is the latest iteration in the YOLO series of real-time object detection algorithms. It is designed to achieve high accuracy and speed in detecting multiple objects in images or video streams.

```python
from ultralytics import YOLO

# Load a model
model = YOLO("yolov8n.yaml")  # build a new model from scratch
model = YOLO("yolov8n.pt")  # load a pretrained model (recommended for training)

# Use the model
model.train(data="coco128.yaml", epochs=3)  # train the model
metrics = model.val()  # evaluate model performance on the validation set
results = model("https://ultralytics.com/images/bus.jpg")  # predict on an image
path = model.export(format="onnx")  # export the model to ONNX format
```



https://github.com/ultralytics/ultralytics