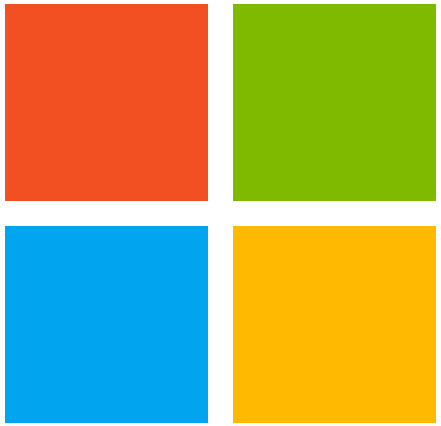


2019 TJMSC Tech. Courses

Computer Vision

Bingchen Zhao
zhaobc@tongji.edu.cn
Tongji Microsoft Student Club

May, 26th
Room 516, Ji Shi Building
Tongji Univ

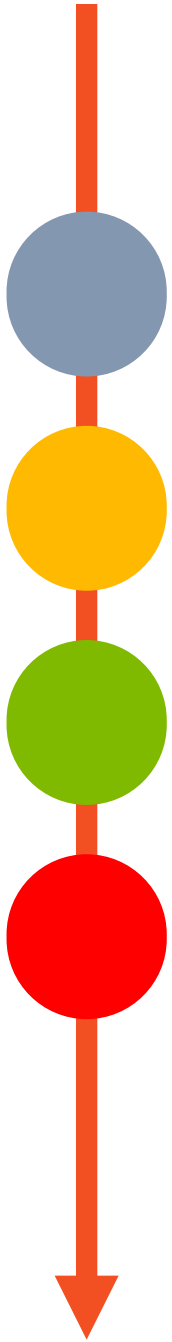


Image representation and classification

Start simple: Linear hypothesis

Non-linearity and feature composition

Interpretations of learned features

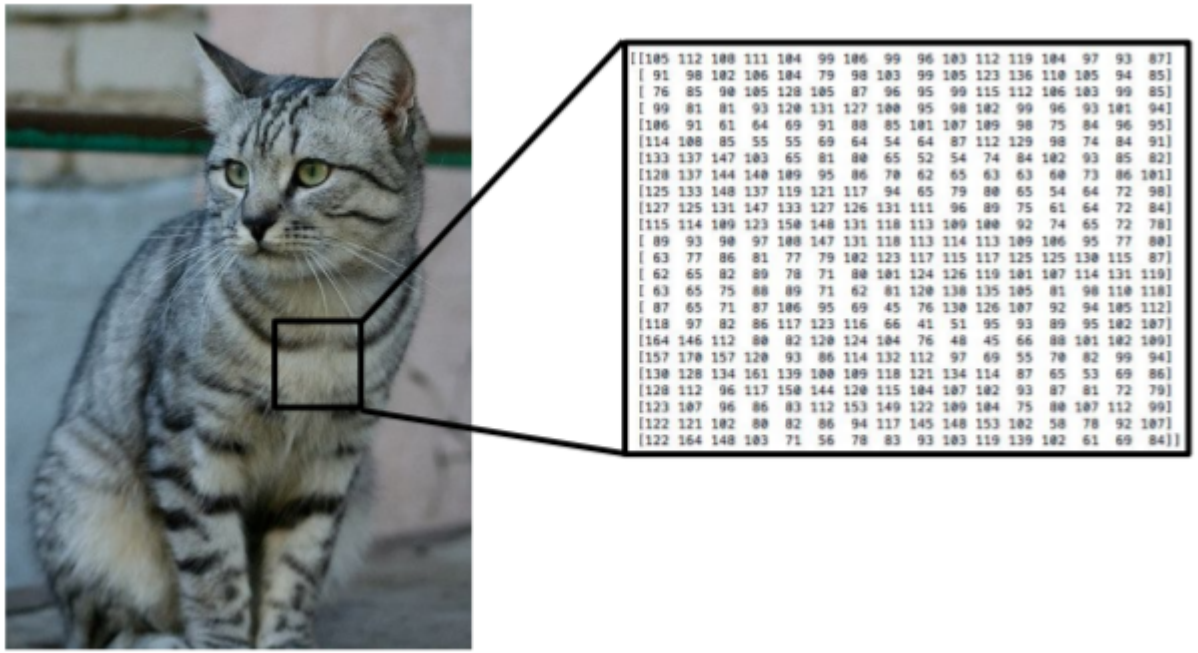
Classification: a core task in CV

f() ⟶Cat!

Maybe sounds easy for human, but hard for computer.

```
def classify_image(image):
    # Implement some good old dark magic here
    return class_label
```

Challenges we are facing: semantic gap



We are seeing a cat, but what computer sees is just a big grid of numbers between [0,255]

Challenges : Background Clutter

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

Challenges : Illumination

This image is CC0 1.0 public domain

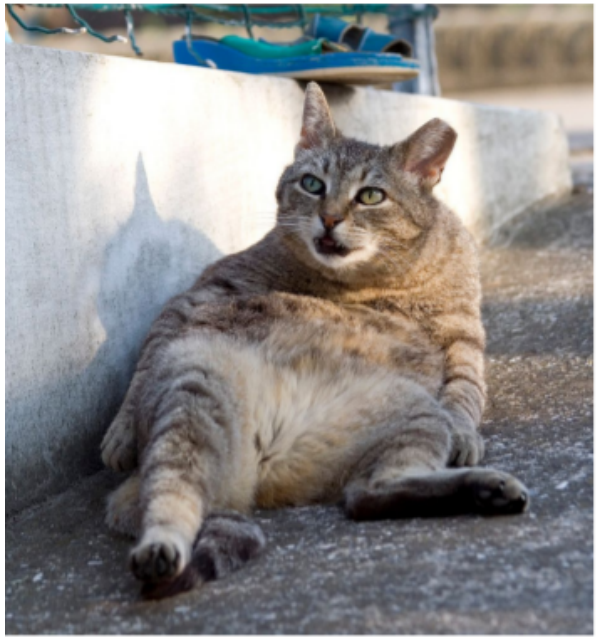
This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

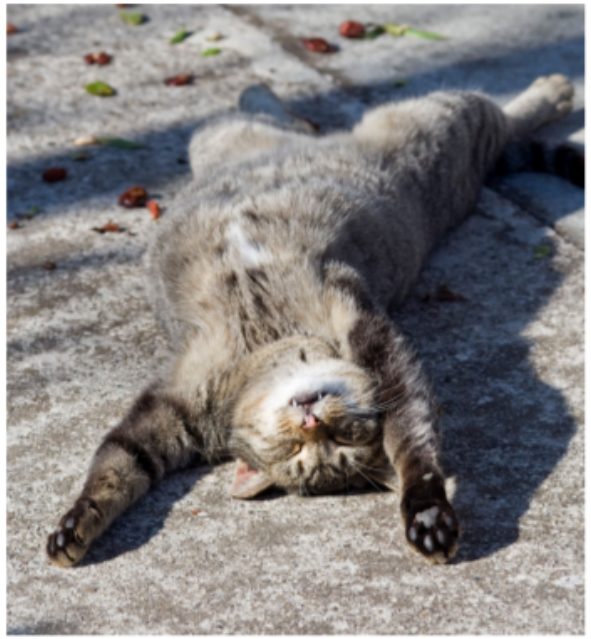
This image is CC0 1.0 public domain

Challenges : Rare pose

同济微软俱乐部
Tongji Microsoft Student Club

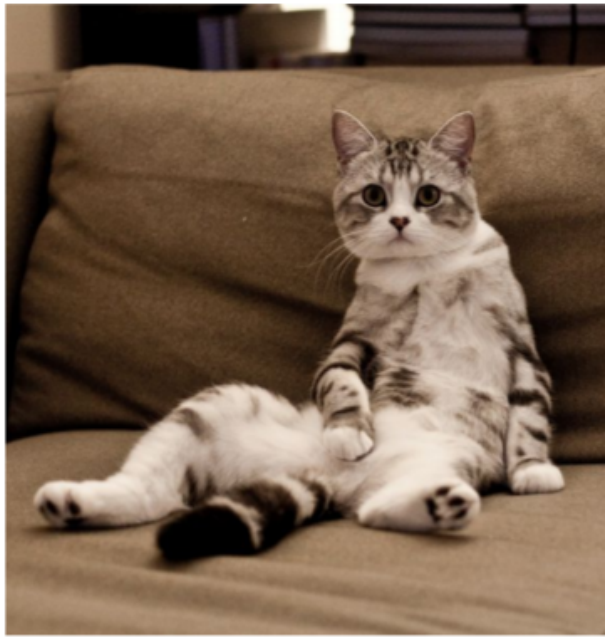


This image by Umberto Salvagnin is licensed under CC-BY 2.0



This image by Umberto Salvagnin is licensed under CC-BY 2.0

This image by sare bear is licensed under CC-BY 2.0



This image by Tom Thai is licensed under CC-BY 2.0

Challenges : Occlusion

This image is CC0 1.0 public domain

This image is CC0 1.0 public domain

This image by jonsson is licensed under CC-BY 2.0

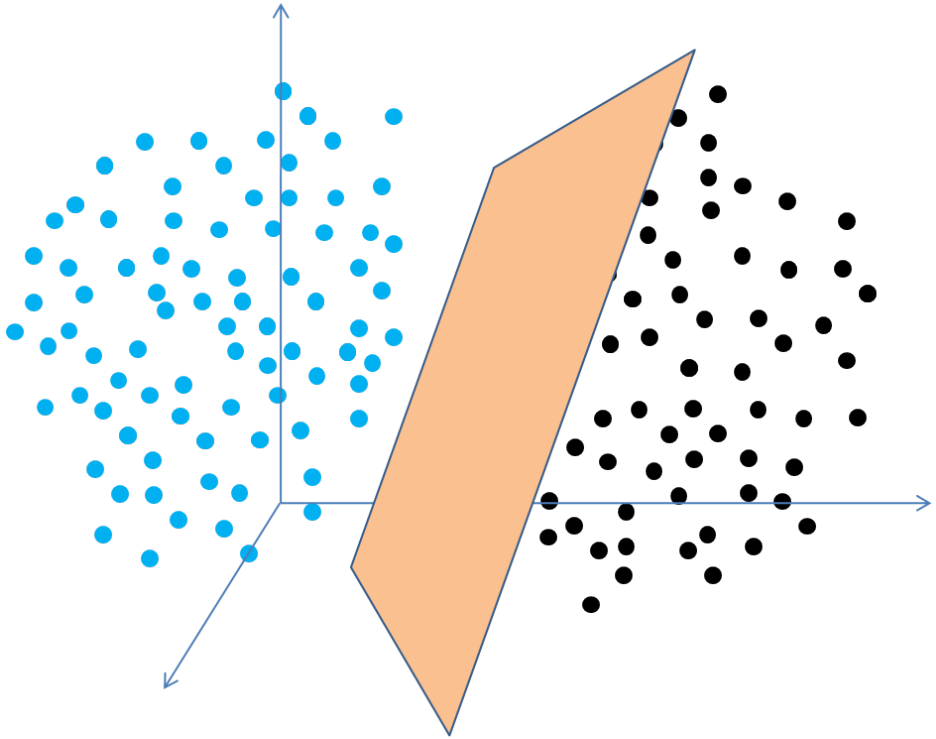
Start simple: CIFAR-10

50,000 training images
each image is **32x32x3**

10,000 test images.

Start simple: Linear hypothesis

Linear hypothesis is that we suppose the data is linear separable

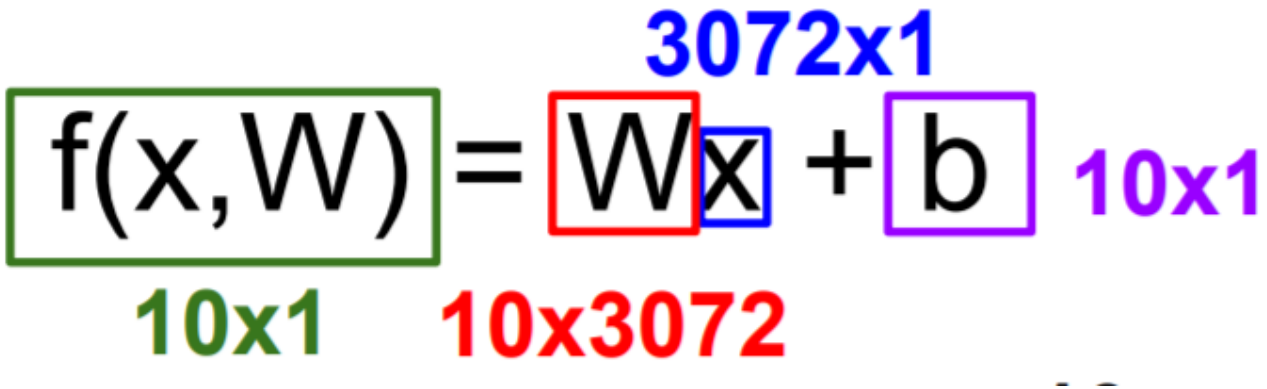


In three dimension, linear separable mean a plane

An image in CIFAR-10 is a array of 32x32x3 numbers(3072 dimension space)

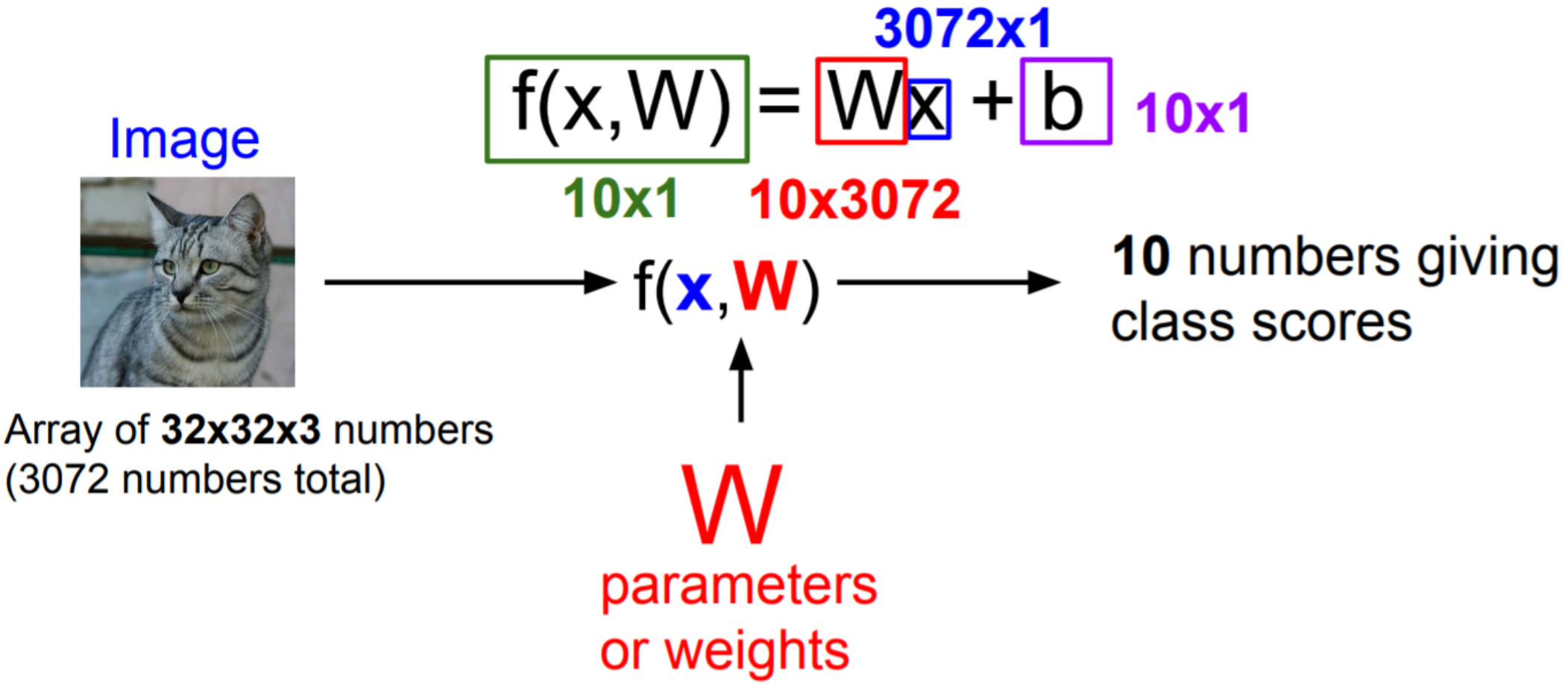
Start simple: Linear hypothesis

Suppose we already know the parameter of the hyper-plane(plane in high dimension), we can use it to define our magical function




```
def classify_image(image, W, b):
    # params:
    #   image is a flattened 3072 array
    #   W and b is our hyper plane parameters
    class_label = image * W + b
    # class_label is predicted class scores
    return class_label
```

Start simple: Linear hypothesis



Start simple: Linear hypothesis

How do we find the parameters for our classifier?

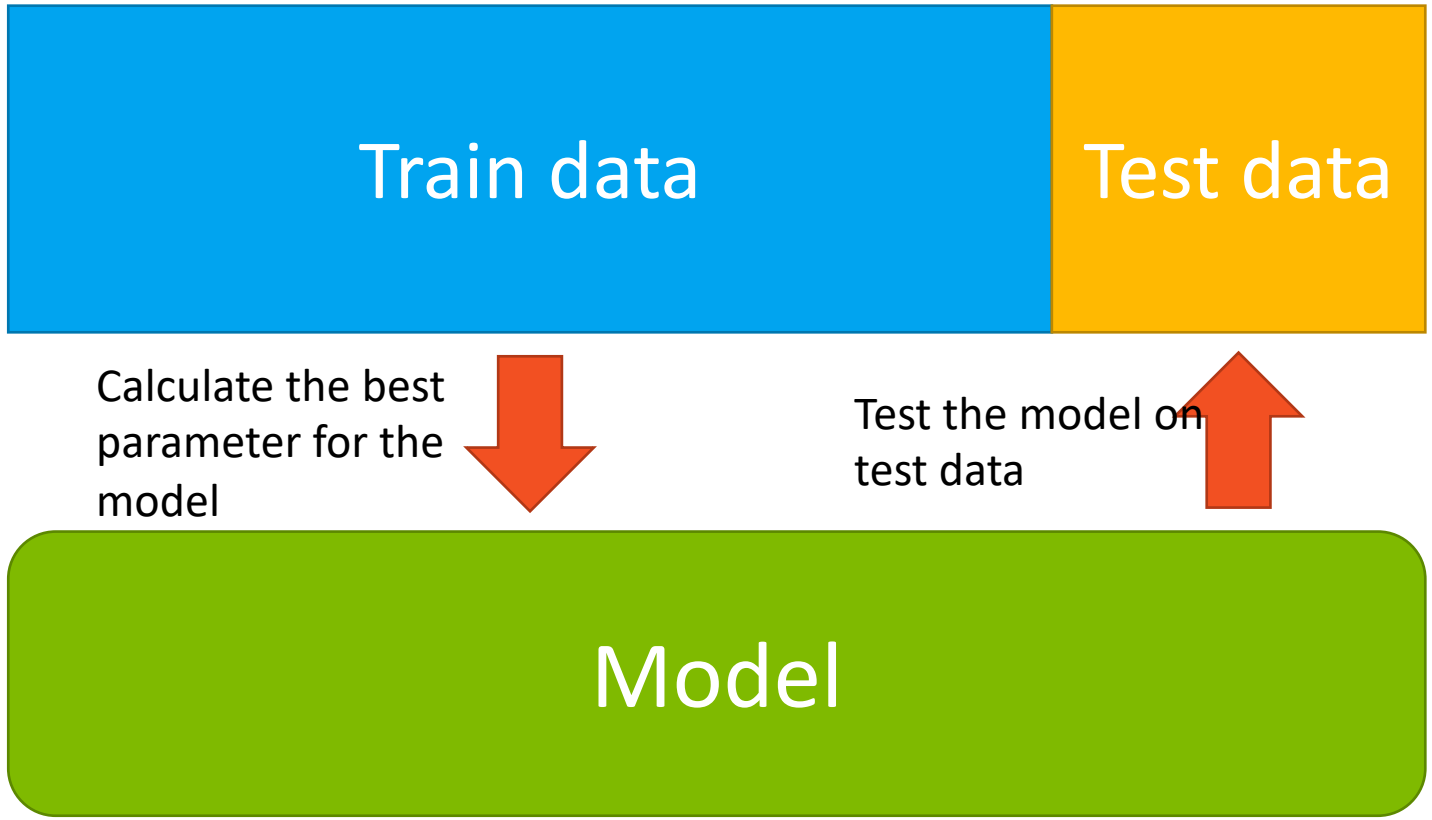
Objective is to find a W minimize the following equation

$$argmin_W \| f(X^{(train)}; W) - y^{(train)} \|_2$$

Least square method gives us

$$W = (X^{(train)T} X^{(train)})^{-1} X^{(train)T} y^{(train)}$$

But this is infeasible cause this need to solve a inverse problem, (sometime you can't)



Start simple: Linear hypothesis

同济微软俱乐部
Tongji Microsoft Student Club

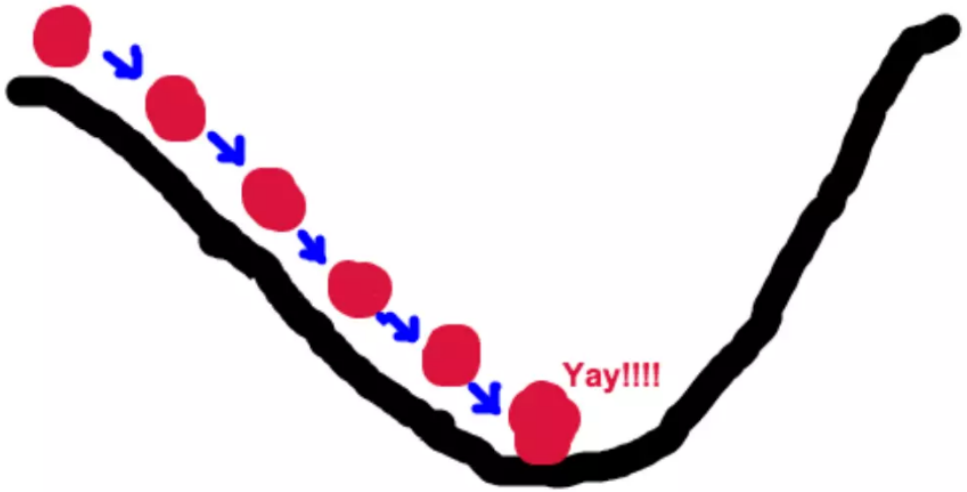
Stochastic Gradient Descent(SGD)

Define the loss function as follows

$$L(W; X, y) = \|f(X; W) - y\|_2$$

Update the W according to the gradient

$$W_{n+1} = W_n - \epsilon \frac{dL}{dW}$$



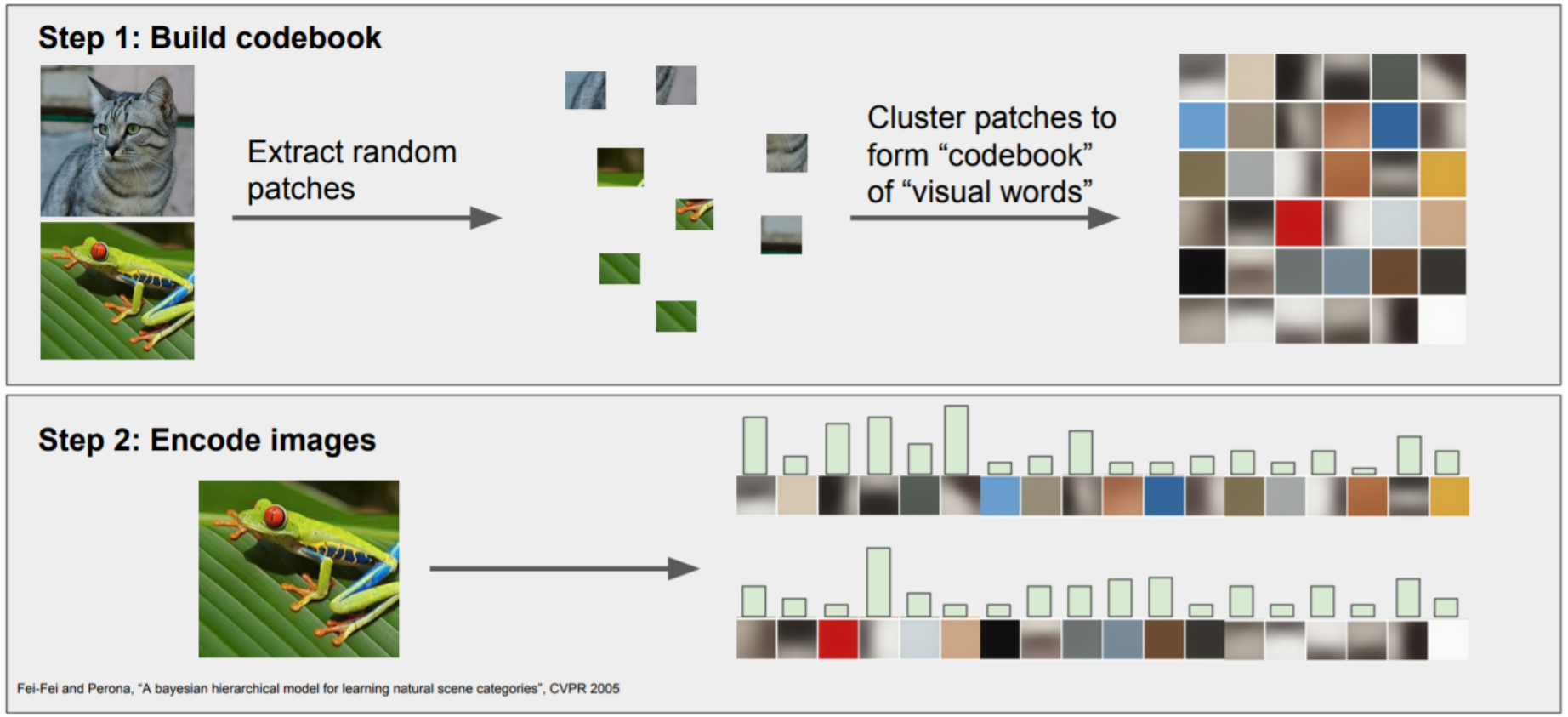
Start simple: Linear hypothesis

Time to put hands on

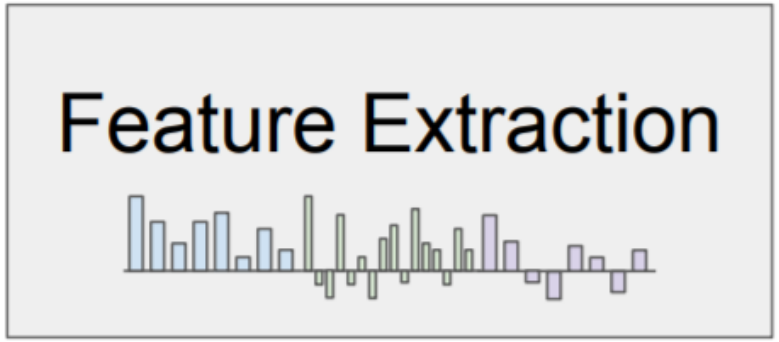
All code and notebook is on Github

Improve the model: feature composition

Previously we are treating every pixel as independent features, what about group nearby pixel together to get more complex features?



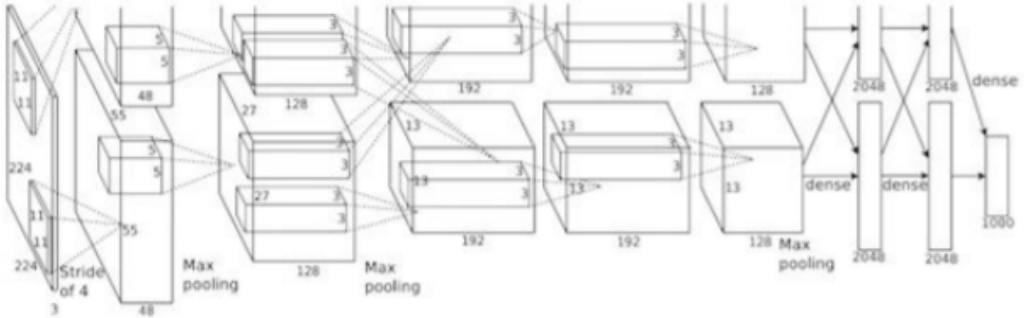
Improve the model: ConvNet



f

10 numbers giving scores for classes

training



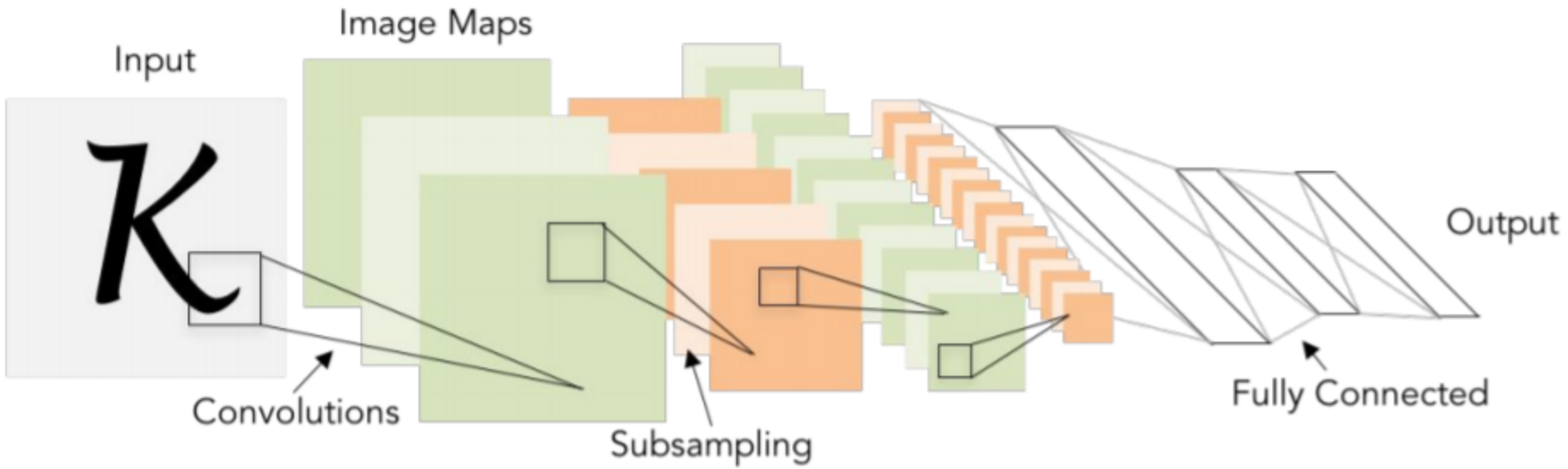
Krizhevsky, Sutskever, and Hinton, "Imagenet classification with deep convolutional neural networks", NIPS 2012. Figure copyright Krizhevsky, Sutskever, and Hinton, 2012. Reproduced with permission.

10 numbers giving scores for classes

training

ConvNet

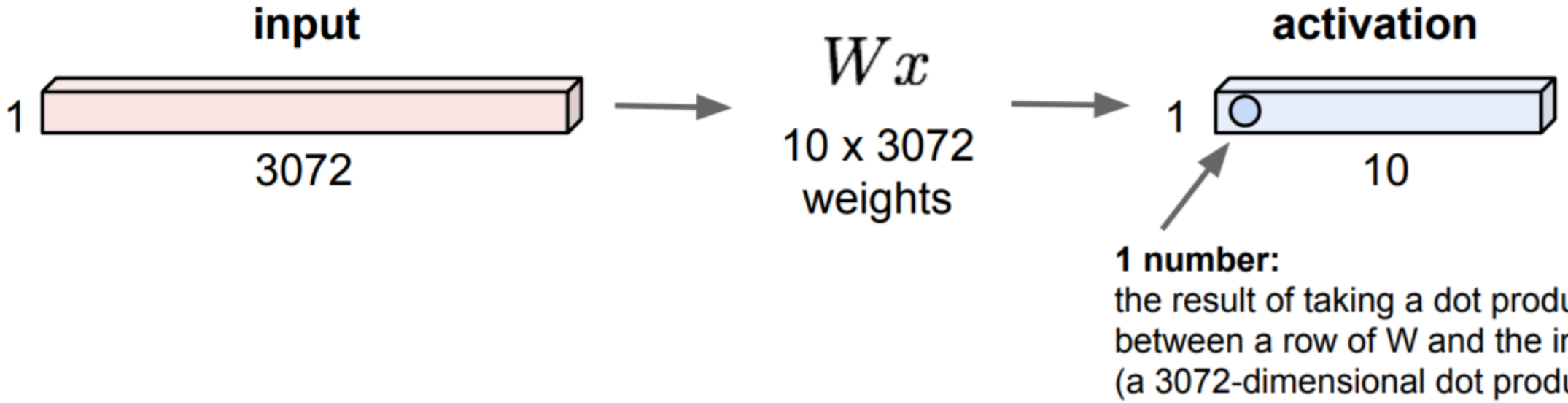
Using Convolutional layer to let the network learn the feature composition, ReLU layer can be used to add non-linearity.



LeNet-5

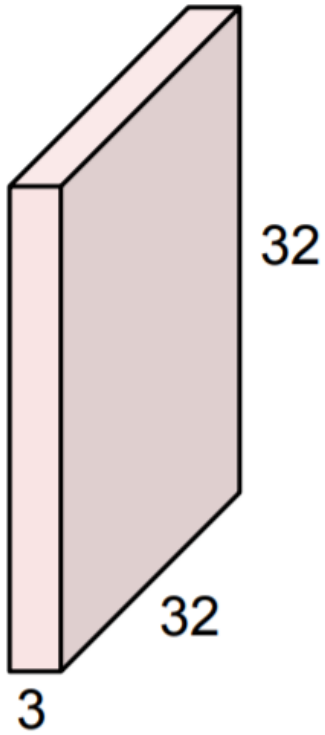
ConvNet

32x32x3 image -> stretch to 3072 x 1

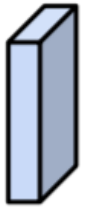


ConvNet

32x32x3 image

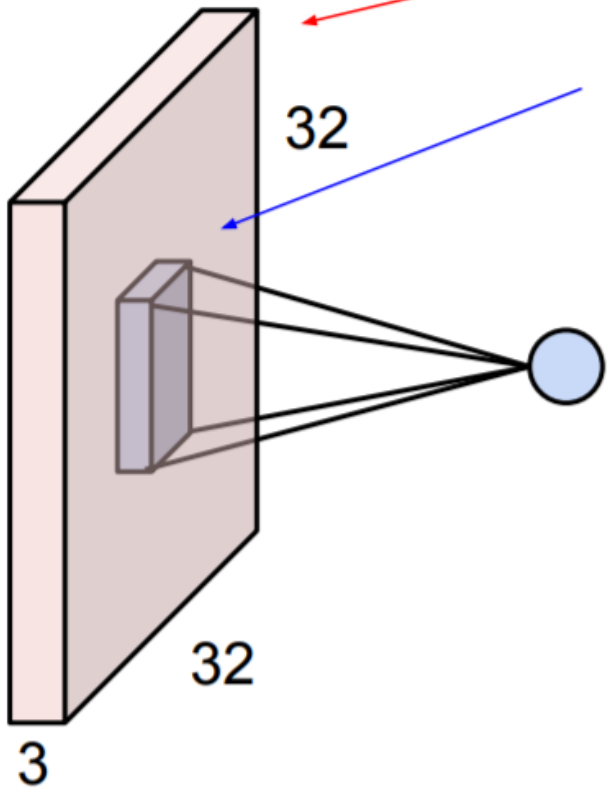


5x5x3 filter



Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

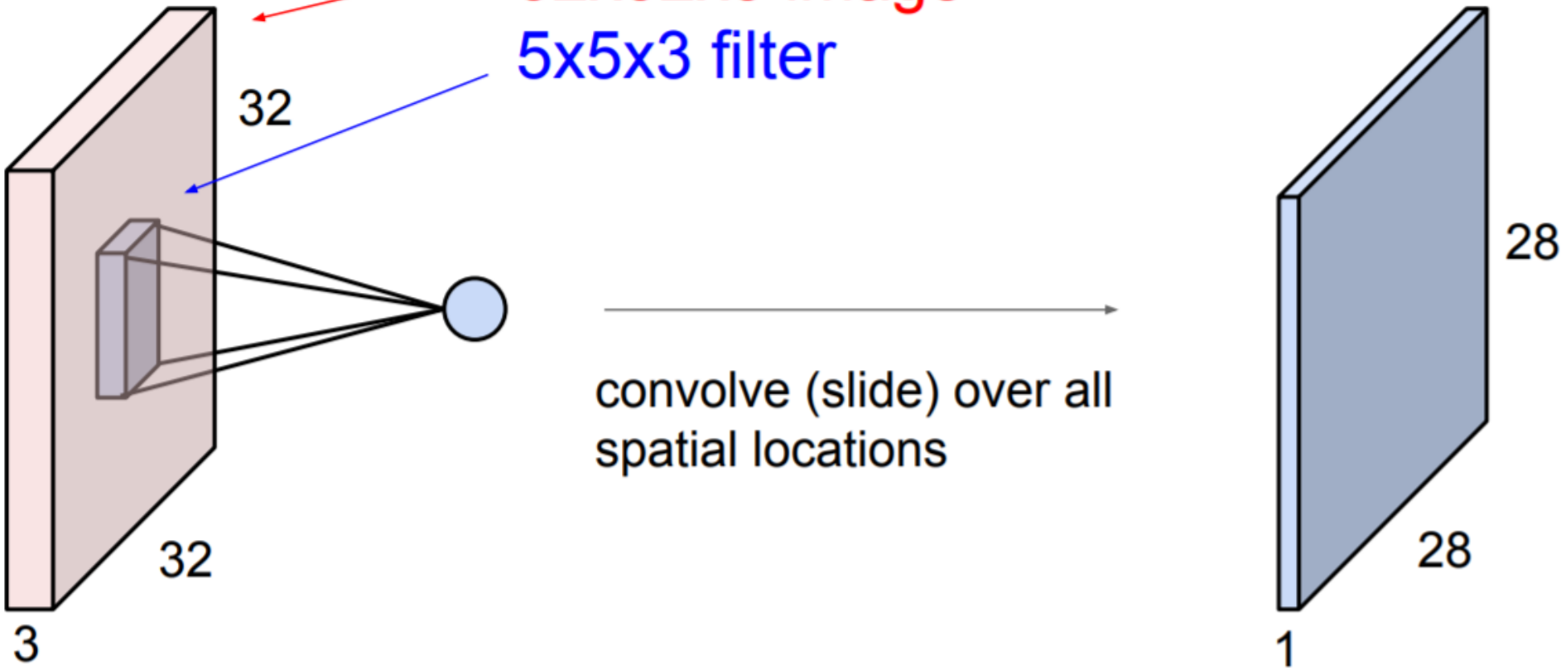
ConvNet



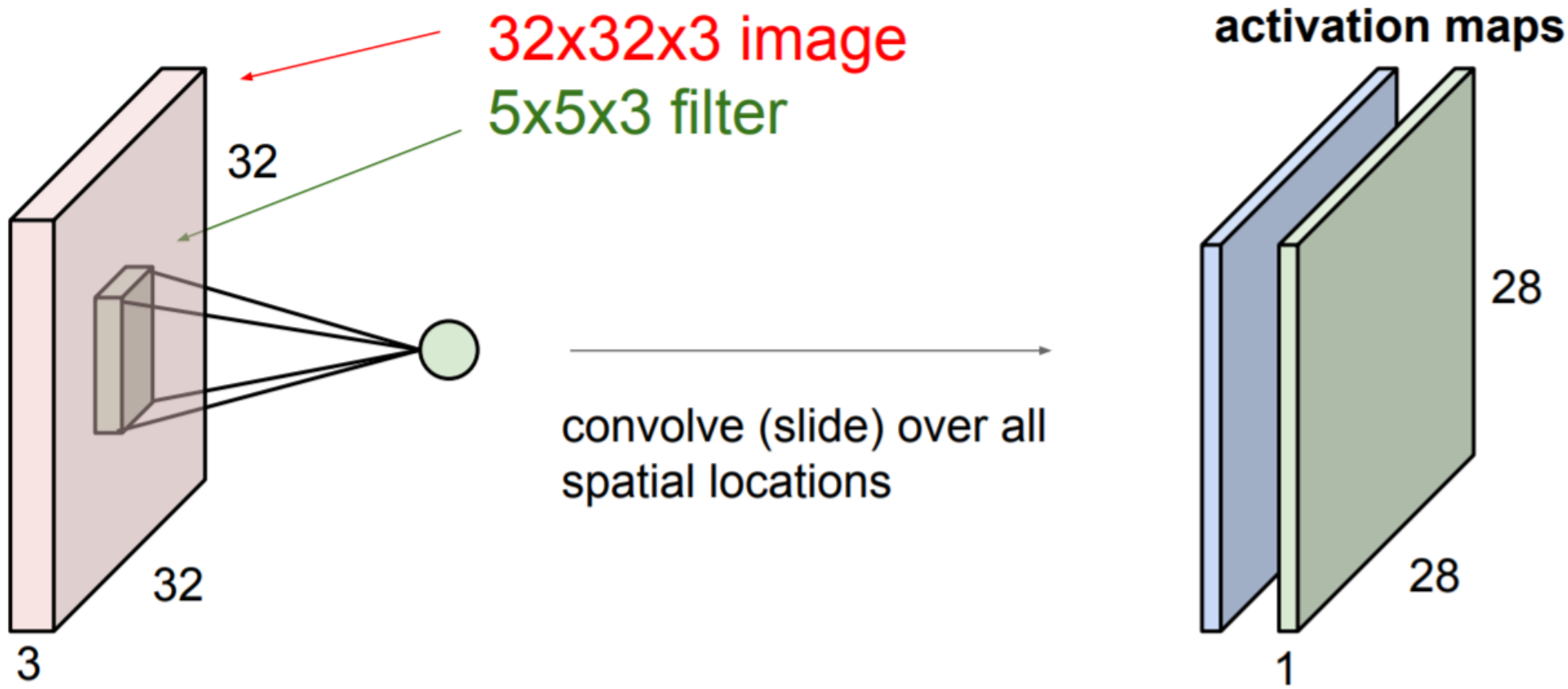
1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

ConvNet

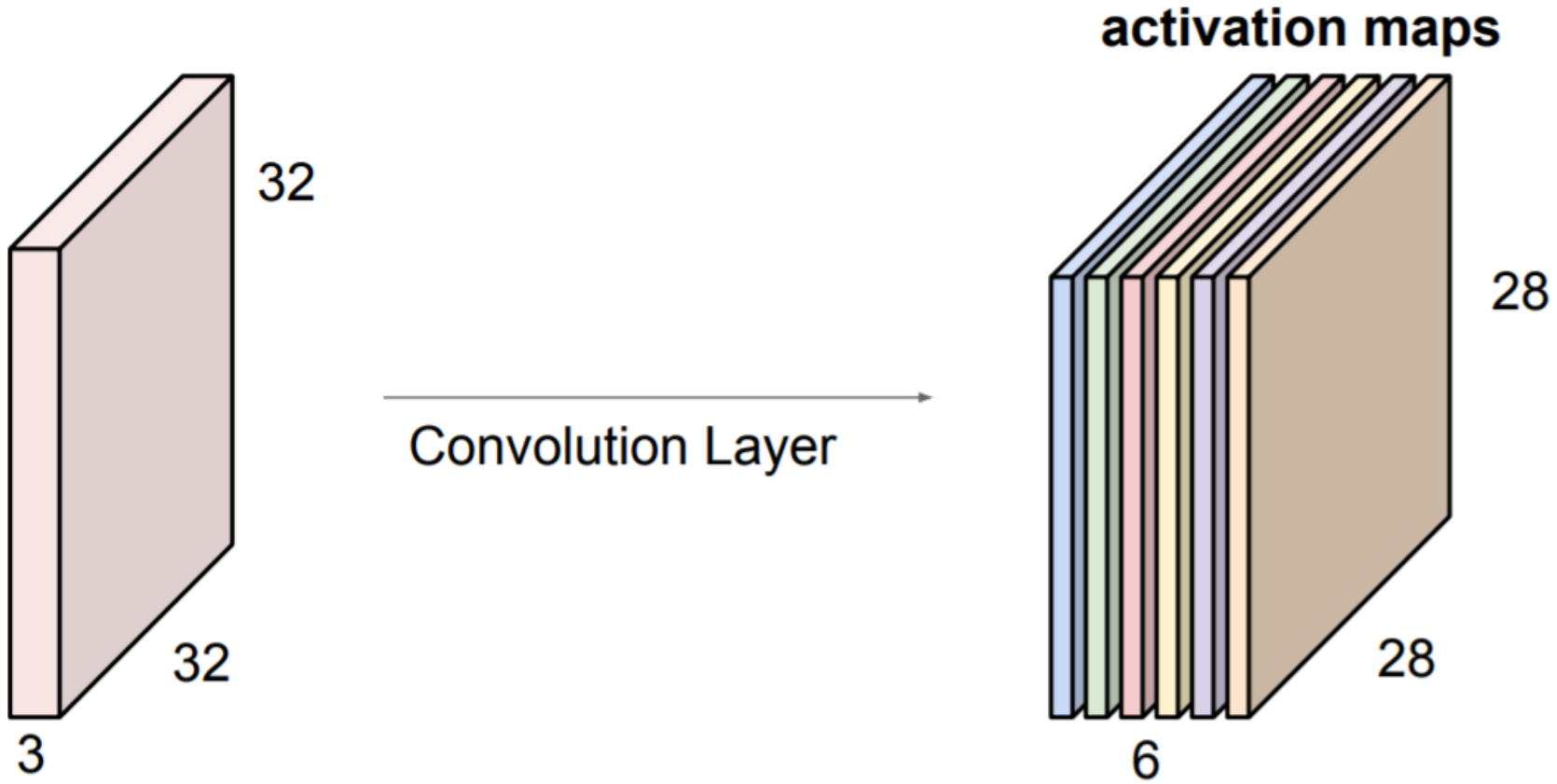

32x32x3 image
5x5x3 filter
32
32
3
convolve (slide) over all
spatial locations
activation map
28
28
1

ConvNet


32x32x3 image
5x5x3 filter
32
32
3
convolve (slide) over all
spatial locations
activation maps
28
28
1

ConvNet

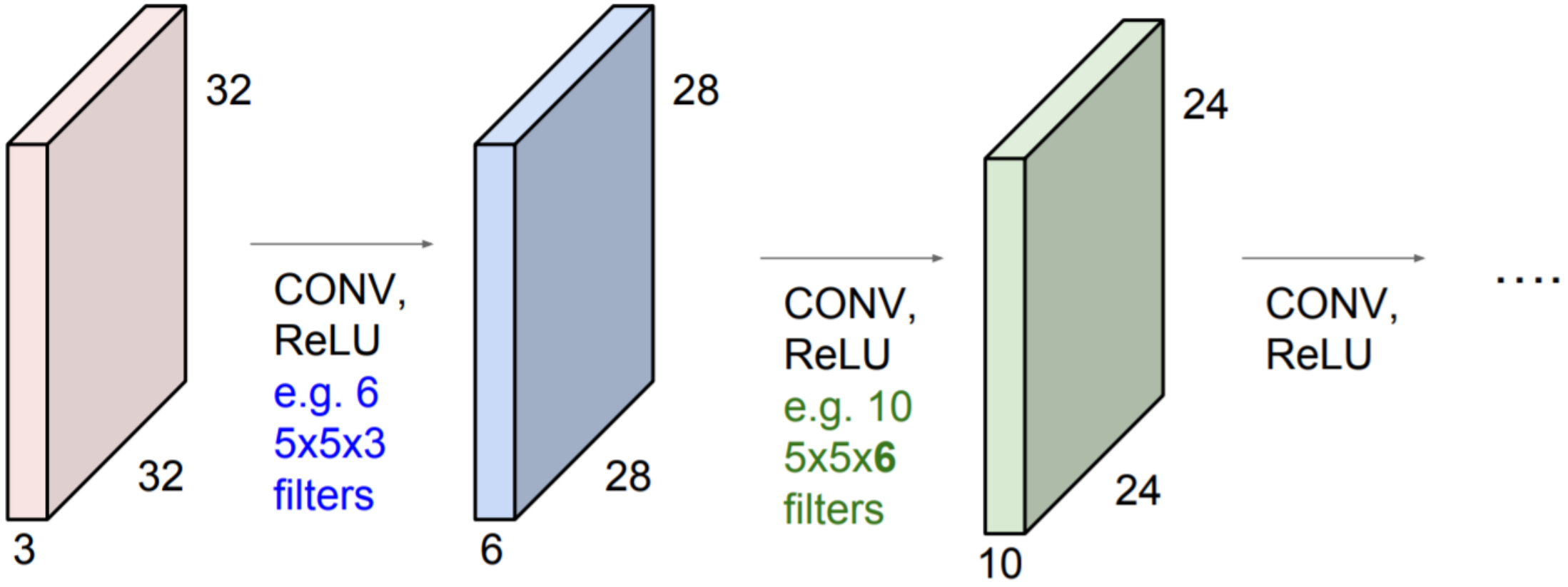
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6!

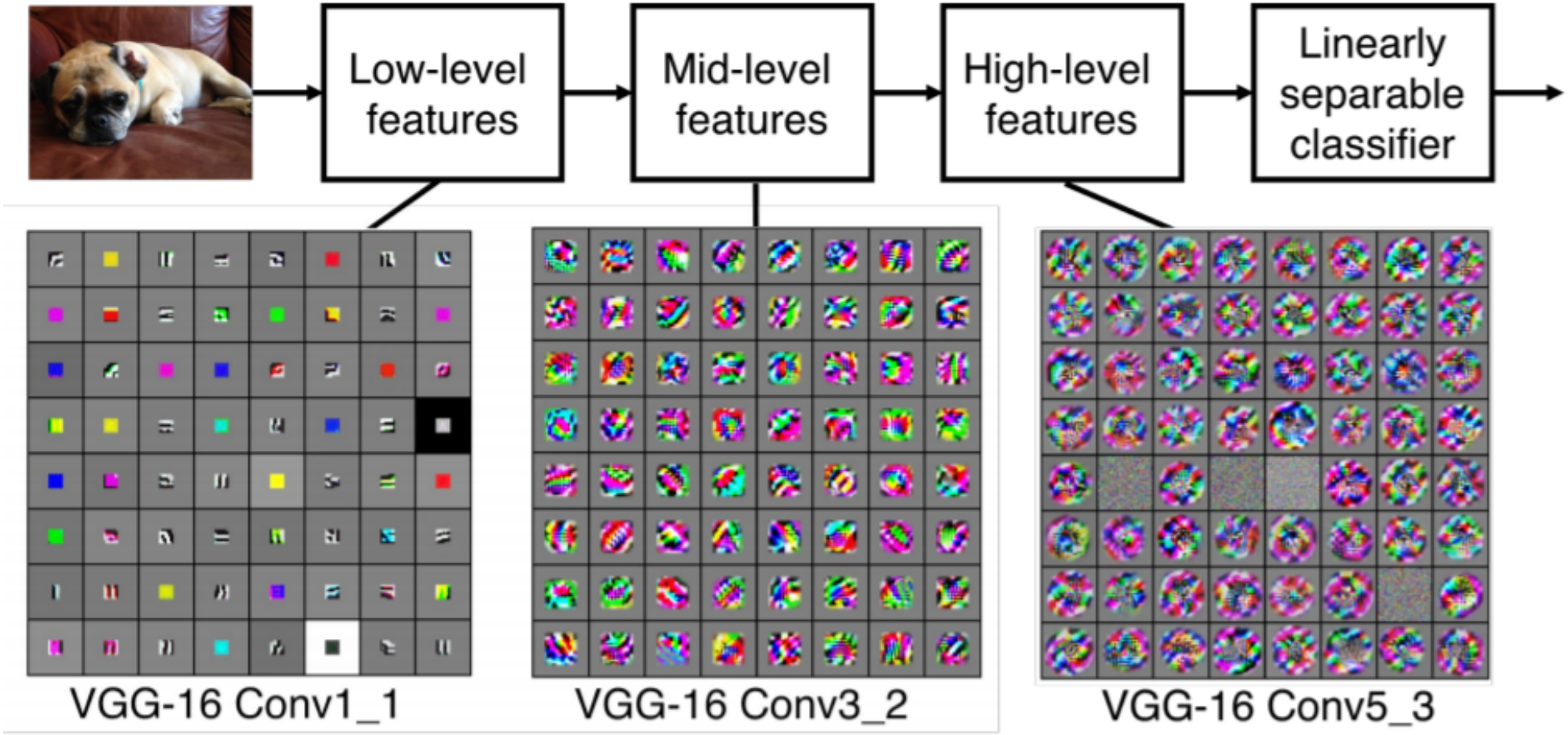
ConvNet

ConvNet is a sequence of Convolution Layers, interspersed with activation functions(ReLU which set value < 0 to 0)



ConvNet

Transfer learning



ConvNet : CIFAR10 Code walkthrough

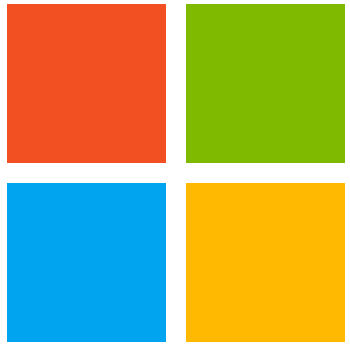

```
Net(
  (conv1): Conv2d(3, 6, kernel_size=(5, 5), stride=(2, 2))
  (conv2): Conv2d(6, 16, kernel_size=(5, 5), stride=(2, 2))
  (fc1): Linear(in_features=400, out_features=120, bias=True)
  (fc2): Linear(in_features=120, out_features=84, bias=True)
  (fc3): Linear(in_features=84, out_features=10, bias=True)
)
```

ConvNet : dog-vs-cat Code walkthrough

同济微软俱乐部
Tongji Microsoft Student Club

ConvNet : Interpretations

ConvNet : Interpretations



Microsoft

同济微软俱乐部
Tongji Microsoft Student Club

加入TJMSC QQ群

关注TJMSC微信公众号