Week 8 | Lecture 09
Convolutional Networks

Wan Fang
Southern University of Science and Technology

Today’s Agenda

- **Convolutional Networks**
  - A Design Challenge in 3D Volumes
  - Convolutional Operation
  - The Design of a Convolutional Layer
- **Layers in ConvNets**
  - Three Stages of a Convolutional Layer
  - Common Architectures of ConvNets
  - CONV, POOL, FC Layers
- **A Universal Workflow**
  - Define the Problem & Dataset Assembly
  - Success Metric & Evaluation Protocol
  - Scale up the Model with Regularization
- **Learning a Transition Model**

Convolutional Networks

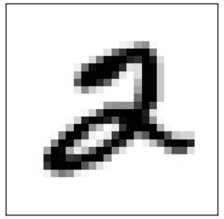
AncoraSIR.com

A Design Challenge with Increasing Dimensions

Regular Neural Nets don't scale well to full images

$\mathbf{512 \times 512 \times 3 = 765,432}$

Gray-scaled



MNIST

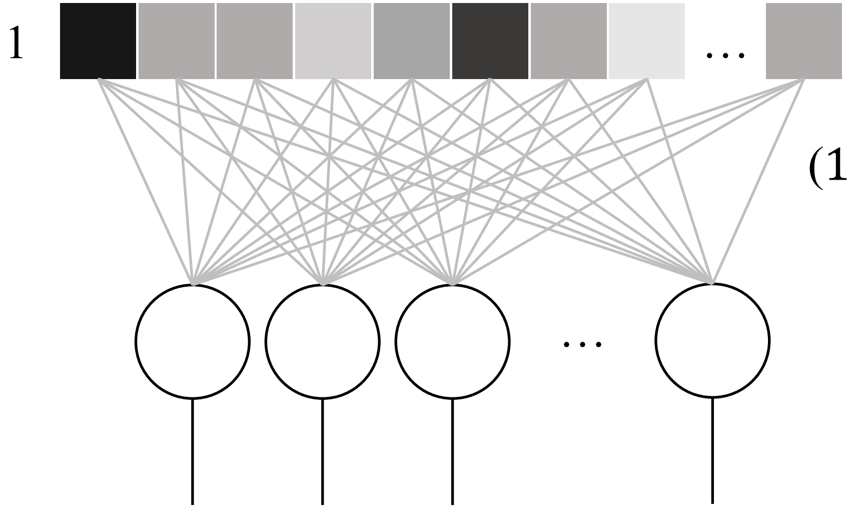
CIFAR-10

Color (RGB)

$28 \times 28 \times 1 = 784$

Input *Features*

$32 \times 32 \times 3 = 3{,}072$



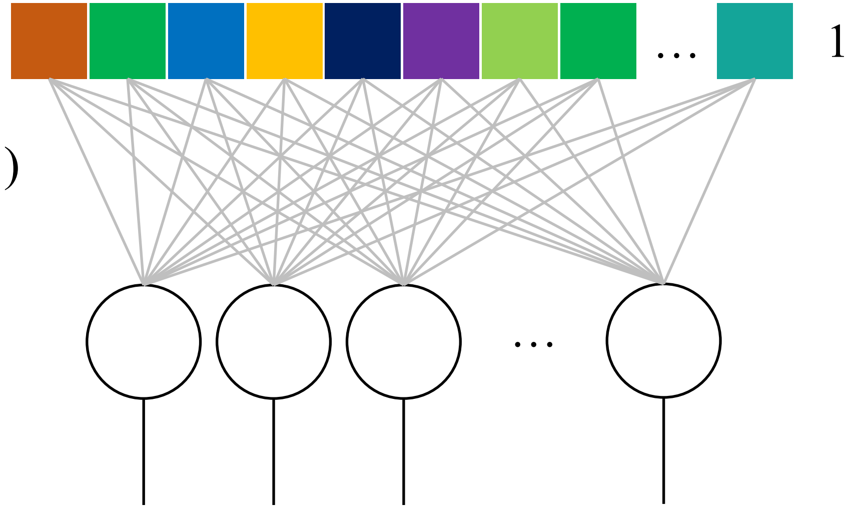
Weighted-Sum

$(10 \times 784 \leftarrow W \rightarrow 10 \times 3072)$

Activation

(?)

Output *Label*



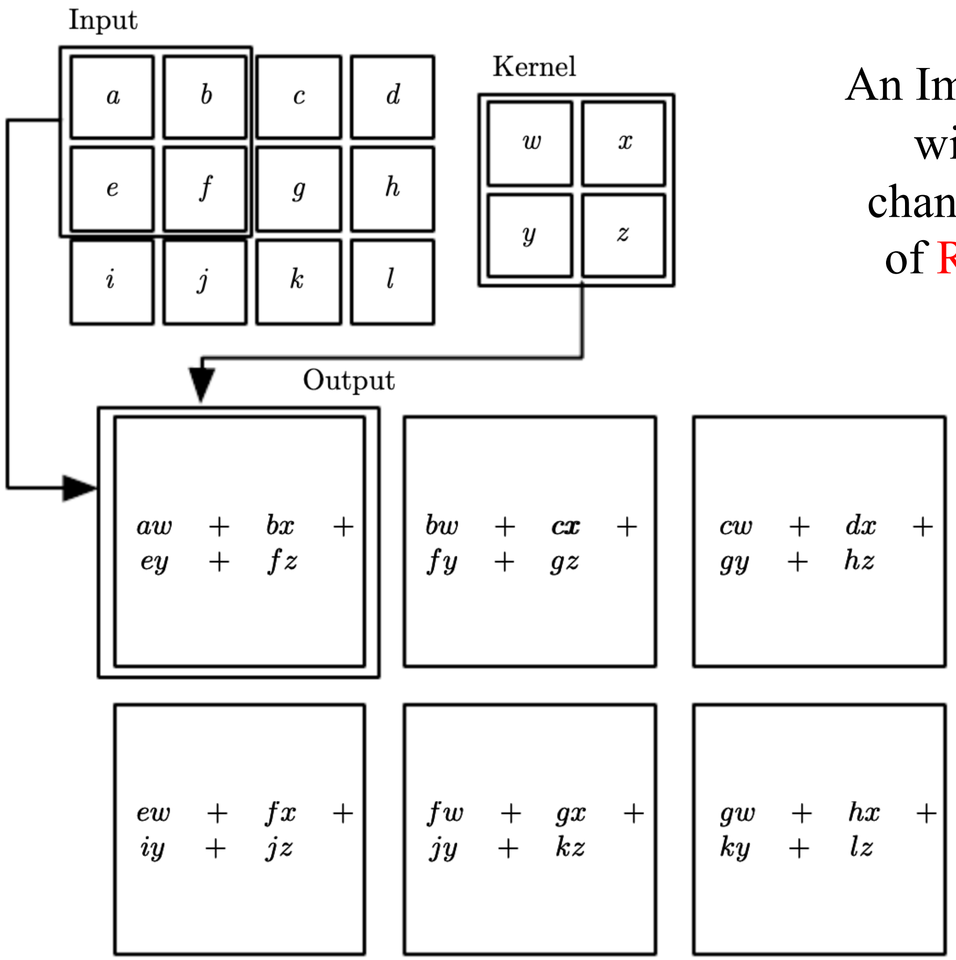
0 1 2 9

airplane cat frog ship

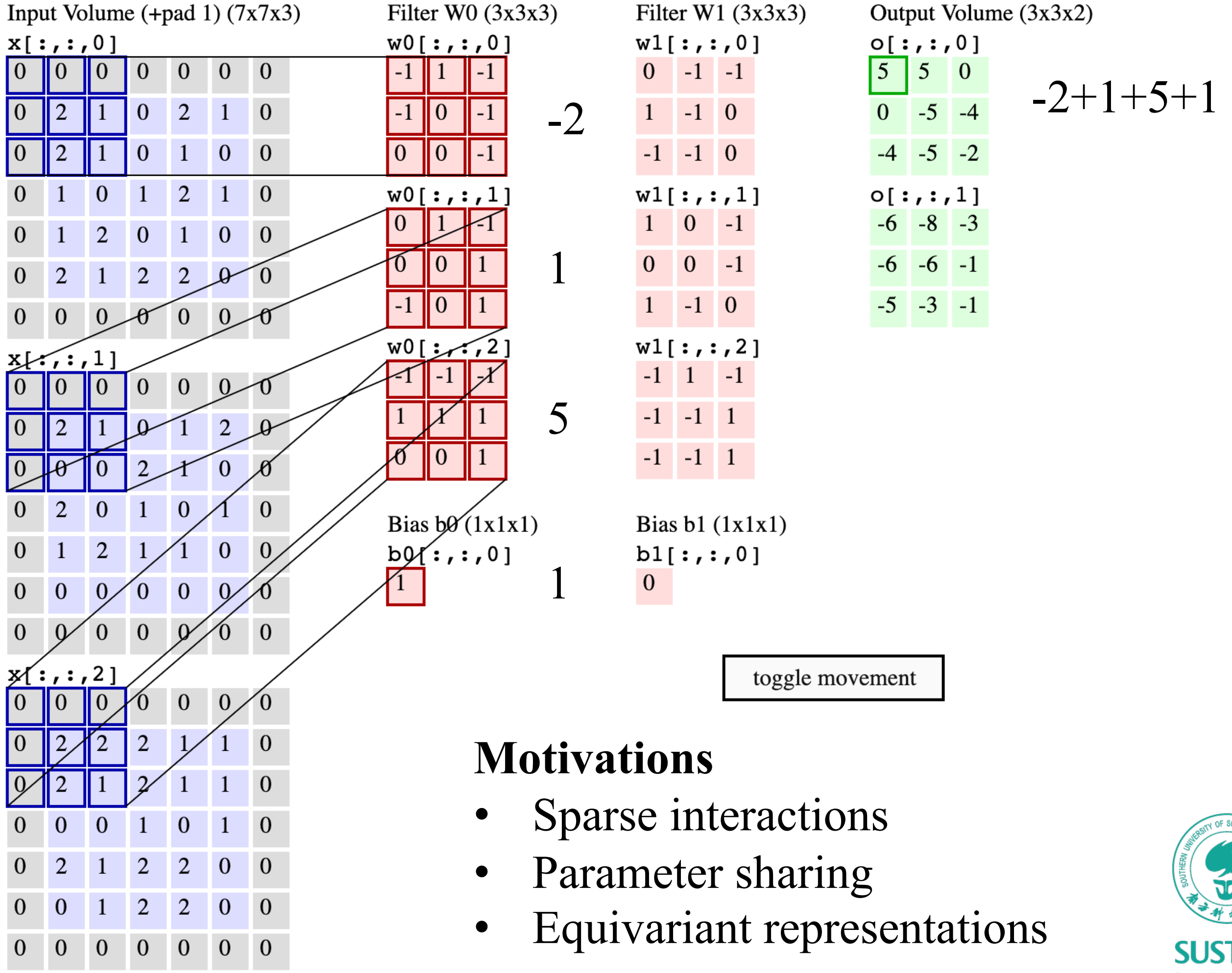
AncoraSIR.com

Convolutional Operation

$$s(t) = \int x(a)w(t-a)da = (x * w)(t)$$



An Image with 3 channels of RGB



Motivations

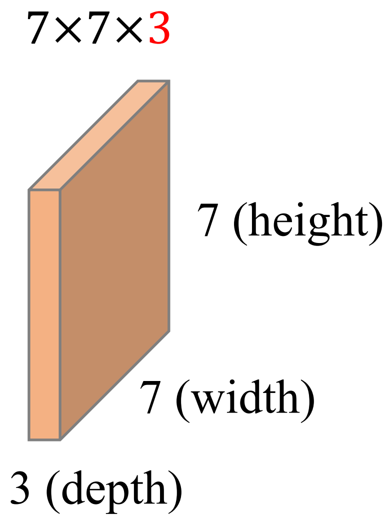
- Sparse interactions
- Parameter sharing
- Equivariant representations

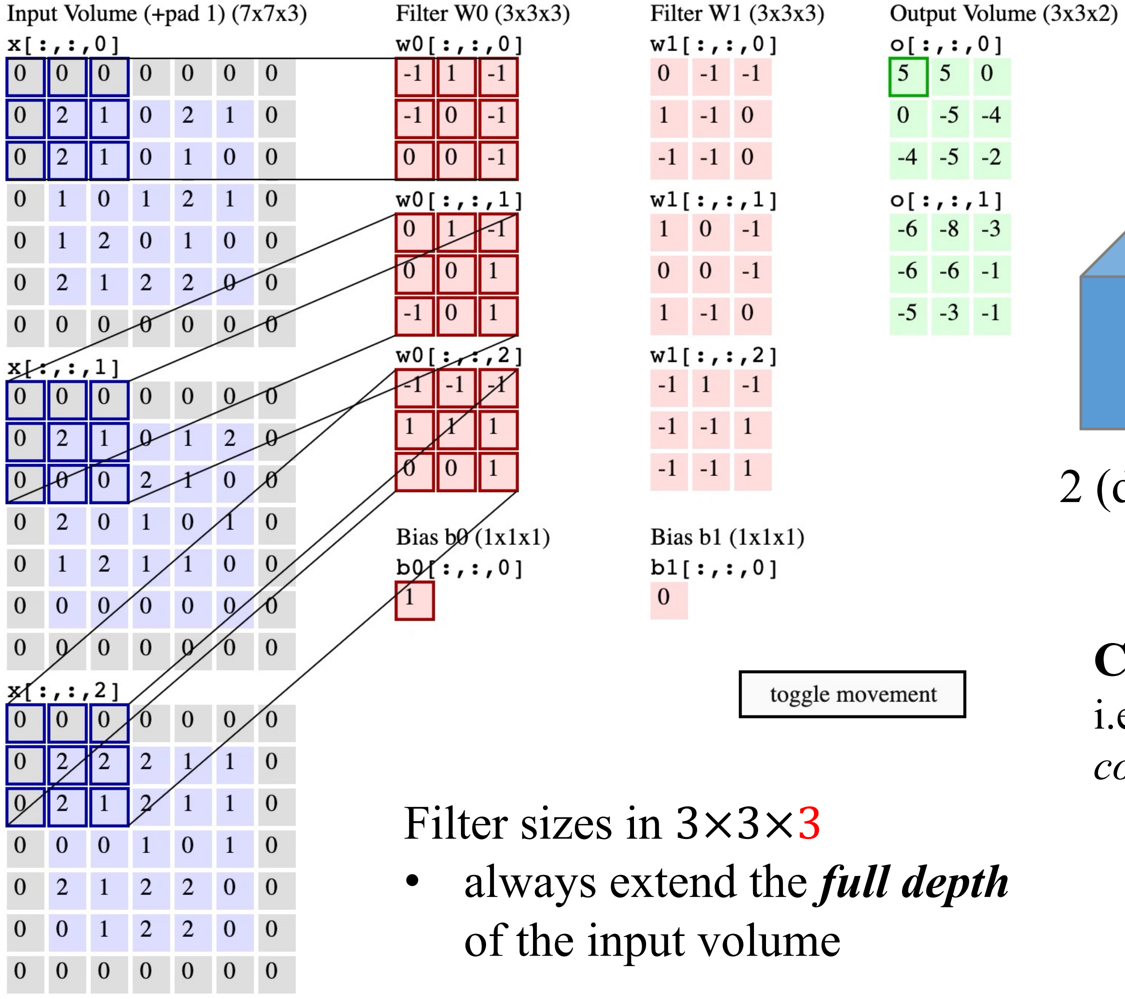
AncoraSIR.com

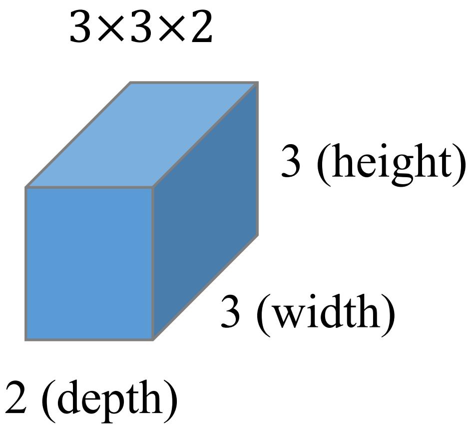
Convolution in 3D Volumes

Preserved spatial structure between the input and output volumes in width, height, number of channels

Input Volume in 3D







Output Volume in 3D

Layers in a ConvNet:
Transform an input 3D volume to an output 3D volume with some differentiable function that may or may not have parameters.

Filter sizes in 3×3×3
- always extend the ***full depth*** of the input volume

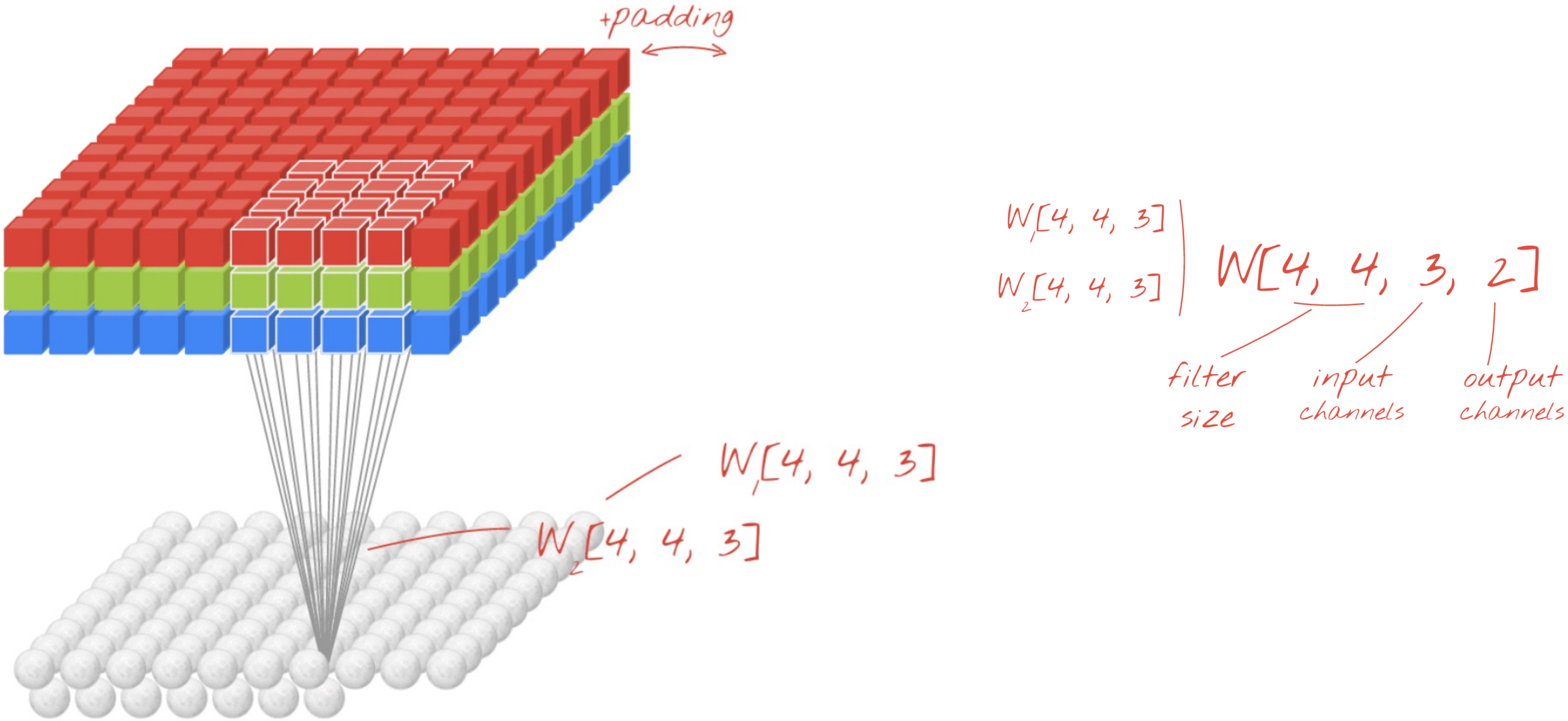
Convolve the filter with the image
i.e. "*slide over the image spatially, computing dot products*"

AncoraSIR.com

SUSTech Southern University of Science and Technology

The Design of a Convolutional Layer

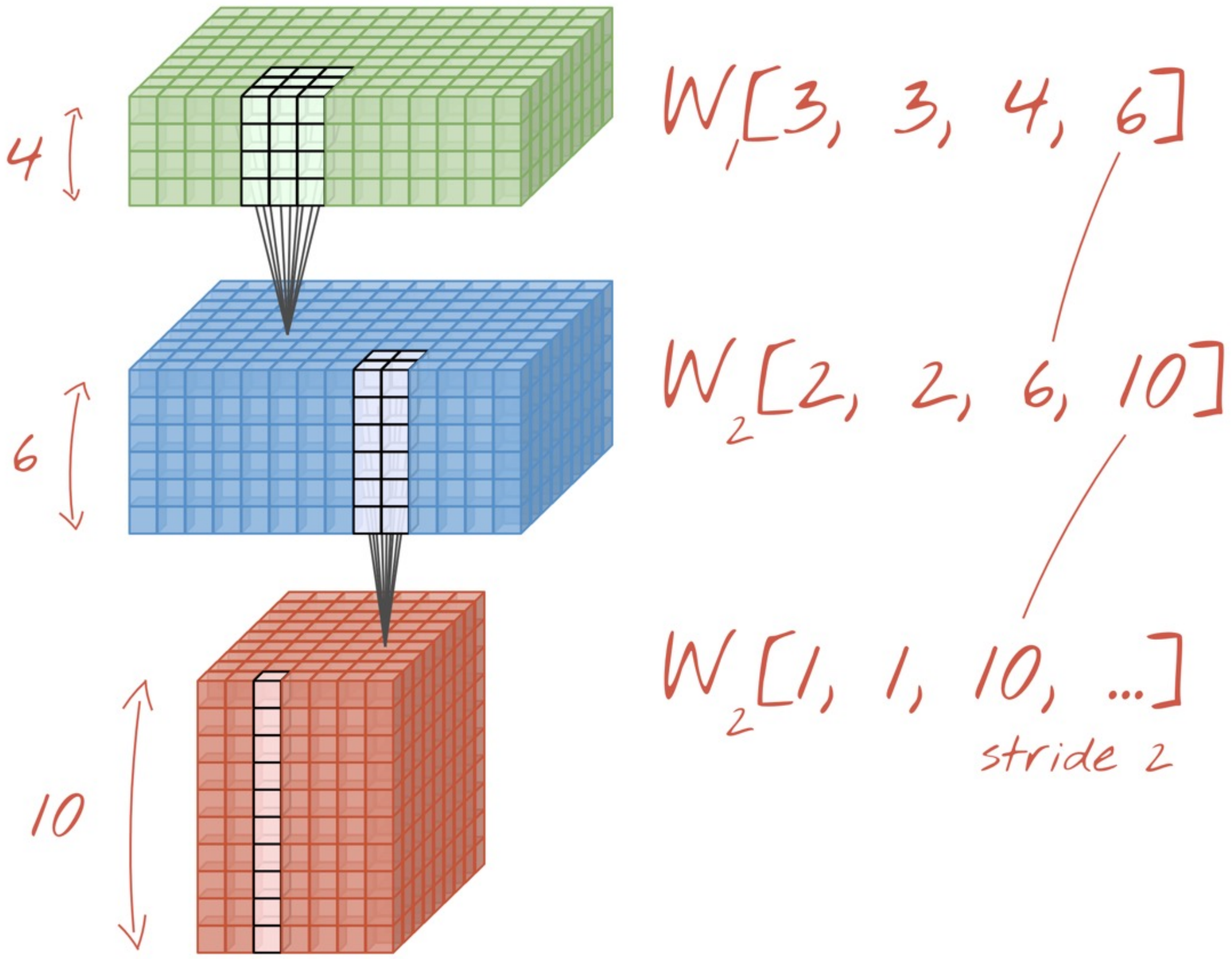
Defined by the filter (or kernel) size, the number of filters applied and the stride



AncoraSIR.com

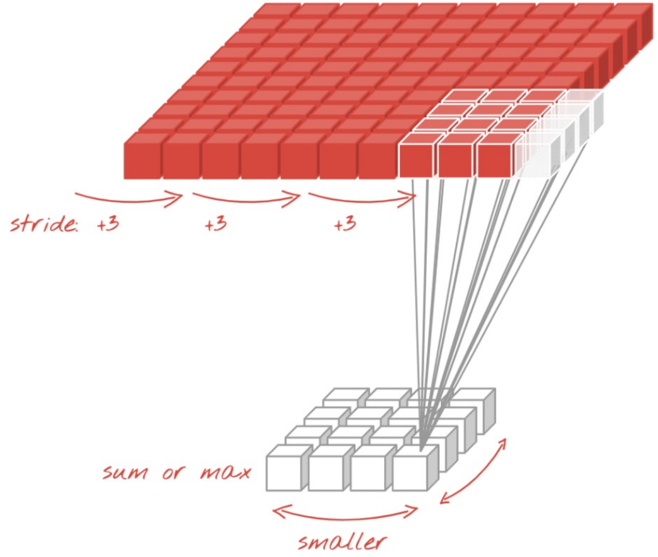
Output Volume Size

Defined by the filter (or kernel) size, the number of filters applied and the stride



AncoraSIR.com

- Depth (number of channels):
  - *adjusted by using more or fewer filters*
- Width & Height:
  - *adjusted by using a stride >1*
  - *(or with a max-pooling operation)*



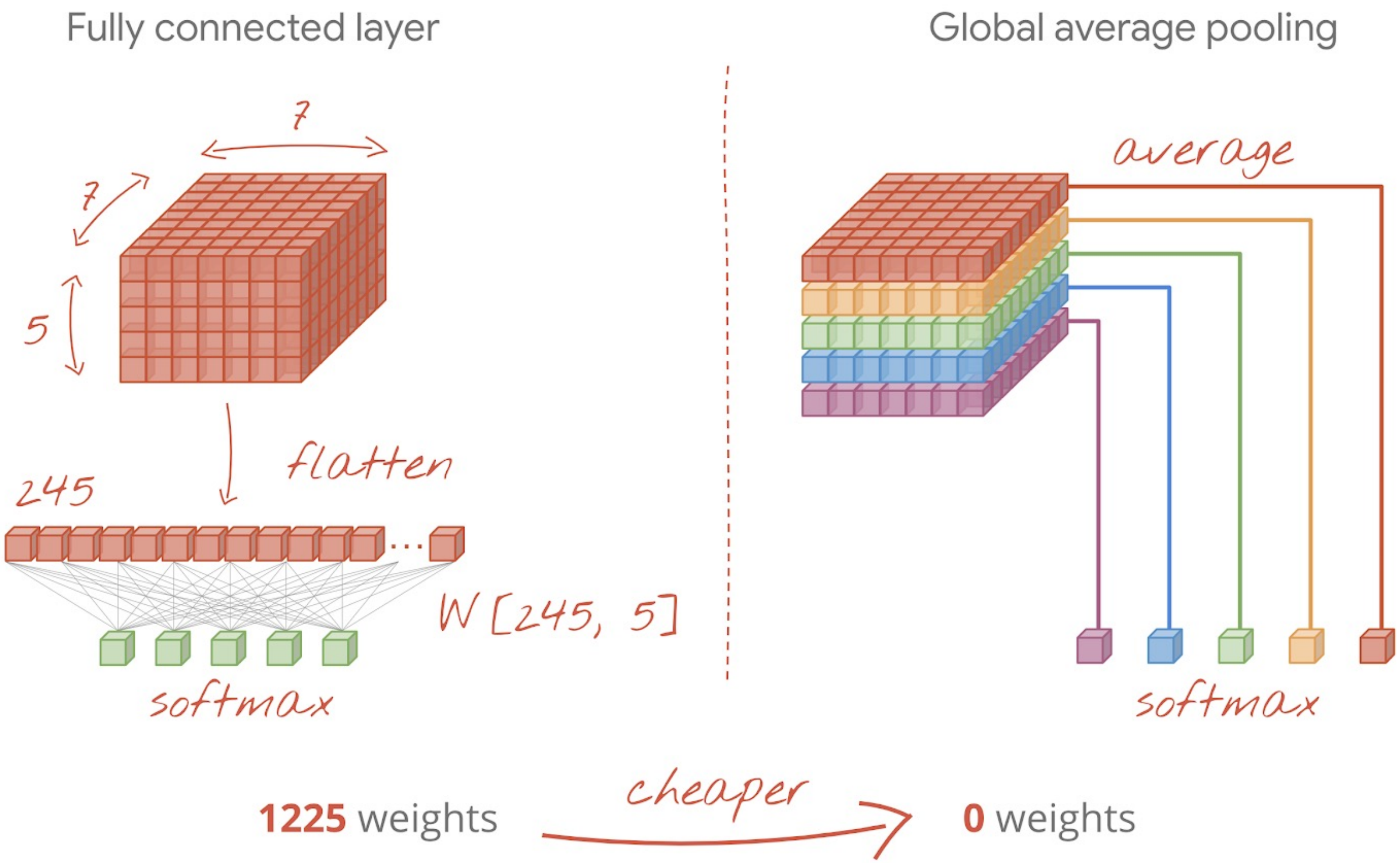
The Last Layer

From a Cubic Volume in 3D to predicted labels

Similar like a normal neural network

Expensive in #weights

But preseves the location data (x, y)



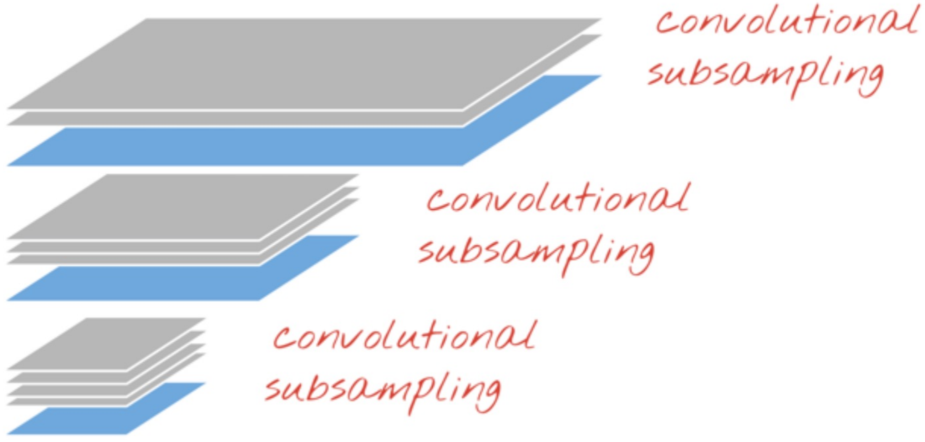
Much lighter in calculation

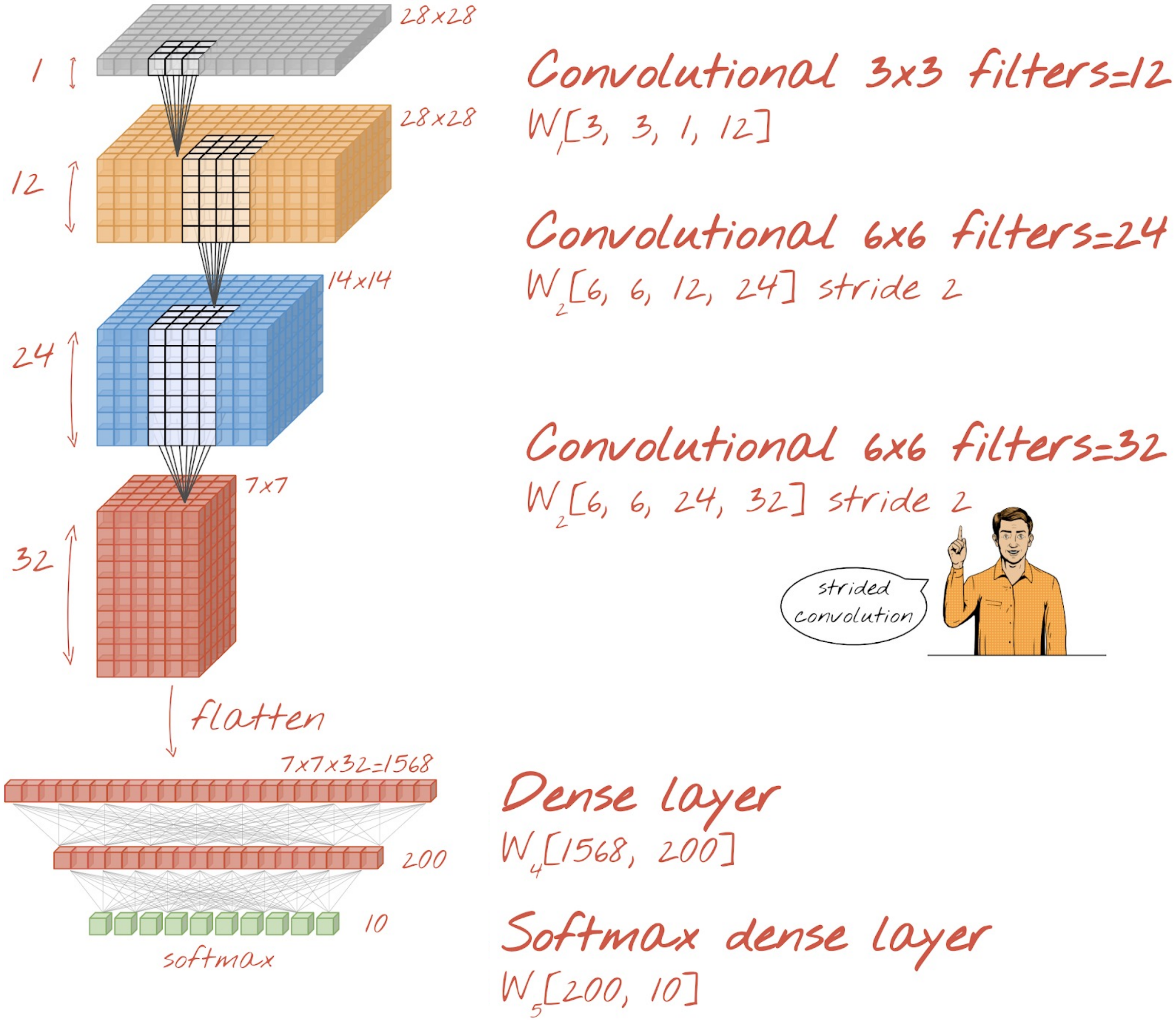
The average pooling explicitly discards all location data

AncoraSIR.com

Stacking Up a ConvNet

Layer-by-layer





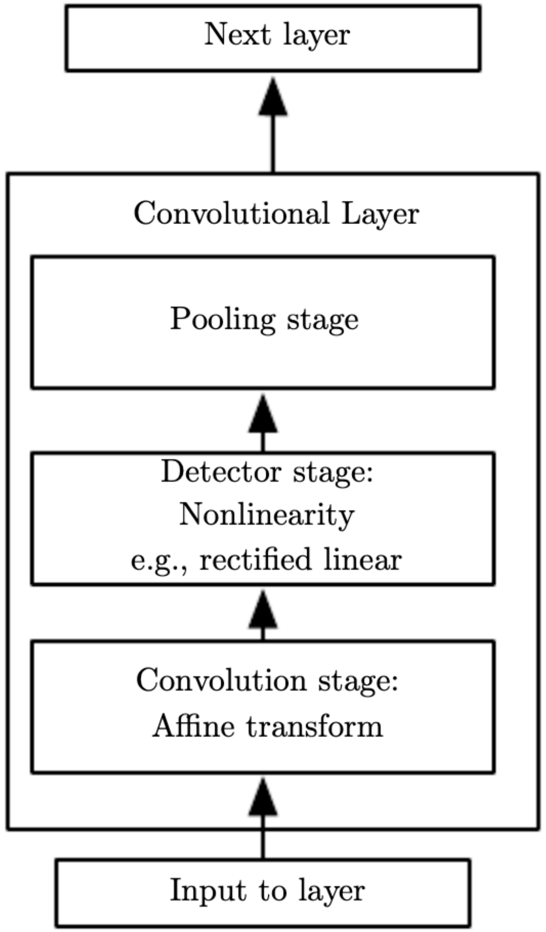
AncoraSIR.com

Layers in ConvNets

AncoraSIR.com

The Three Stages of a Typical ConvNet Layer

The Convolution, Detector and Pooling Stages



AncoraSIR.com

- The maximum output within a rectangular neighborhood (max-pooling)
- The average of a rectangular neighborhood
- The L2 norm of a rectangular neighborhood
- A weighted average based on the distance from the central pixel

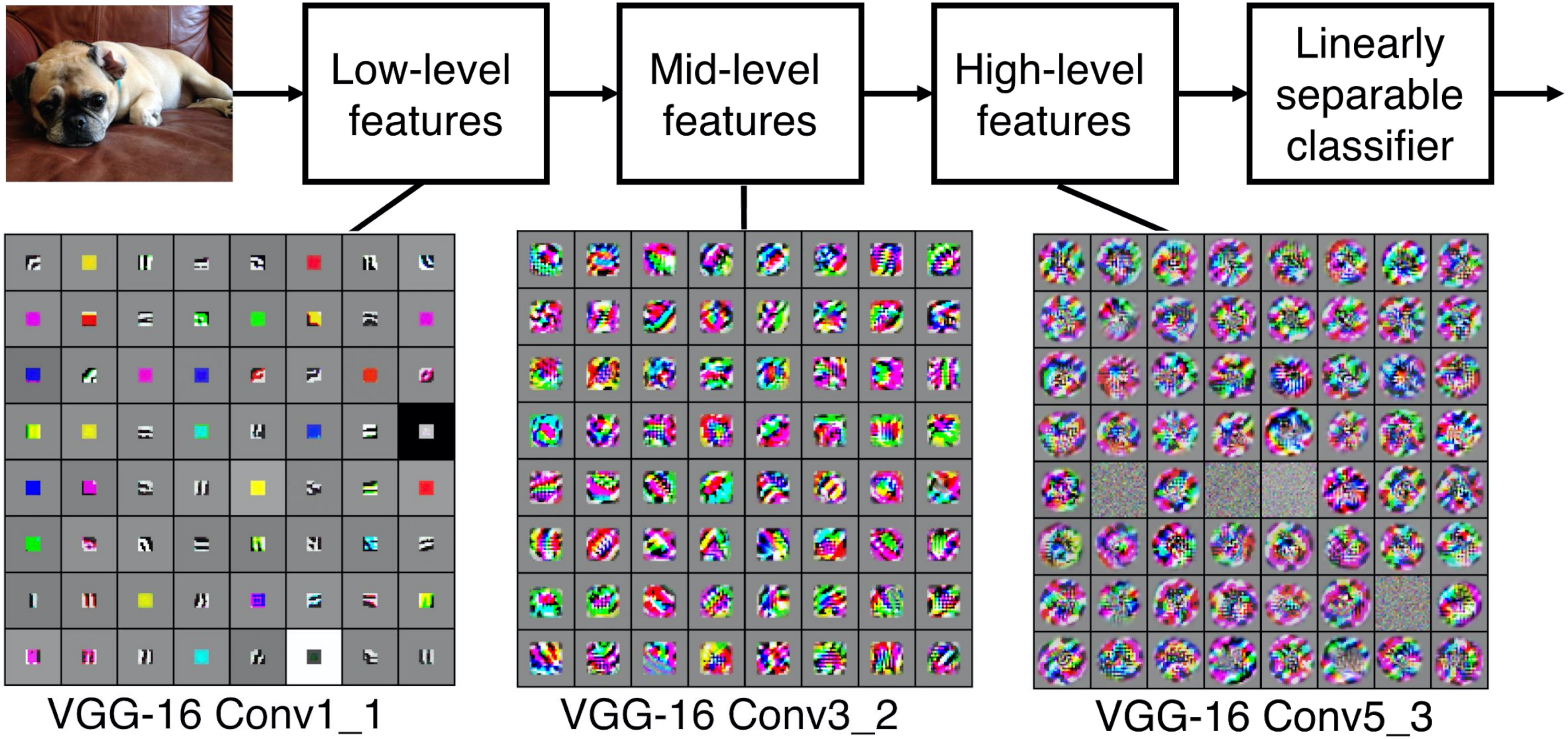
Replace the output of the net at a certain location with a summary statistic of the nearby outputs (can be viewed as a further abstraction of the learned features)

Each linear activation is run through a nonlinear activation function, such as ReLU (can be viewed as activation function)

Performs several convolutions in parallel to produce a set of linear activations (can be viewed as weighted-sum)

A Visualized Understanding of ConvNet

Multi-layered abstraction of 3D features towards a linerly separable classification



AncoraSIR.com

https://arxiv.org/pdf/1311.2901.pdf

A Simple ConvNet for CIFAR-10 Classification

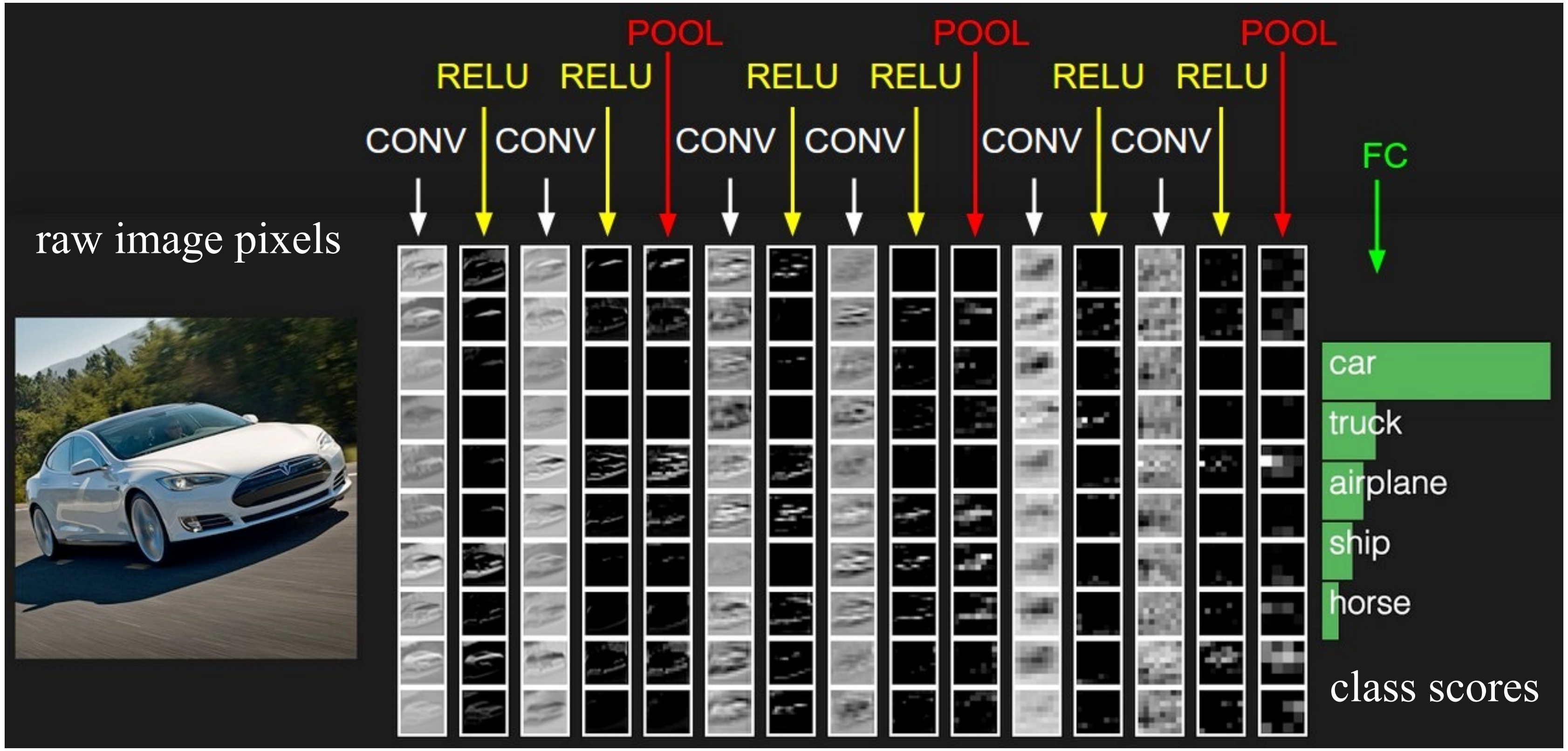
[INPUT - CONV - RELU - POOL - FC]

CONV layer compute the output of neurons that are connected to local regions in the input, i.e. [32x32x12] with 12 filters.

RELU layer will apply an elementwise activation function, such as the max(0,x) thresholding at zero. This leaves the size of the volume unchanged ([32x32x12]).

POOL layer will perform a downsampling operation along the spatial dimensions (width, height), resulting in volume such as [16x16x12].

INPUT layer [32x32x3] will hold the raw pixel values of the image



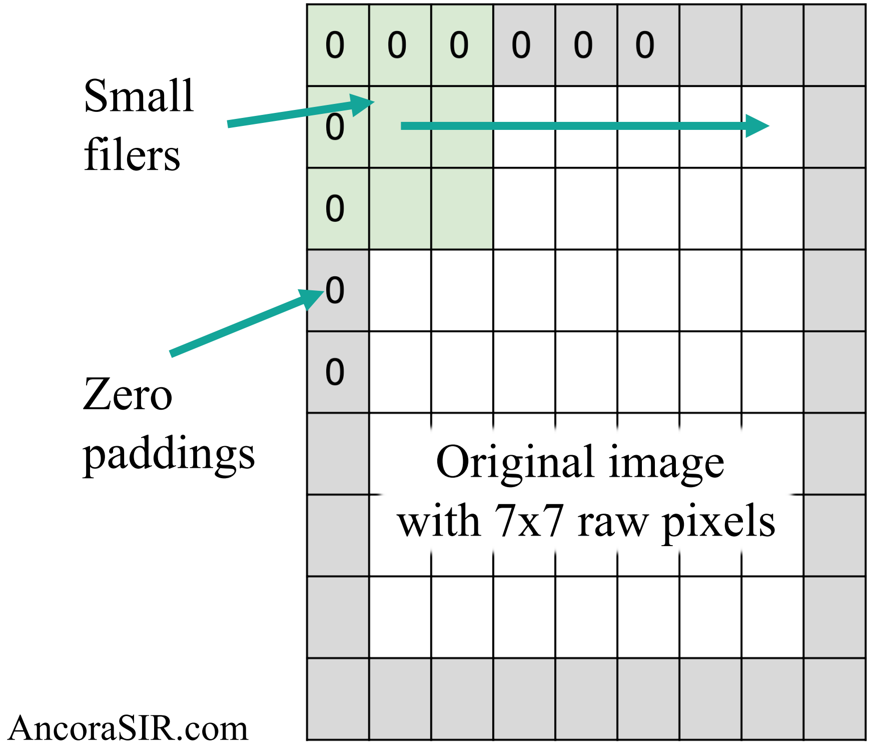
FC (i.e. fully-connected) layer will compute the class scores, resulting in volume of size [1x1x10], where each of the 10 numbers correspond to a class score

AncoraSIR.com

Convolutional Layer

Small filters that slide across the input volume

- Small-size filers
  - e.g. 3x3 or at most 5x5, using a stride of S=1,
  - Padding the input volume with zeros to avoid altering the spatial dimensions of the input.



INPUT features: 7x7

Filer size: 3x3

Stride: 1 (move step-by-step)

Padding: 1 pixel of 0 on all borders

OUTPUT features: 7x7

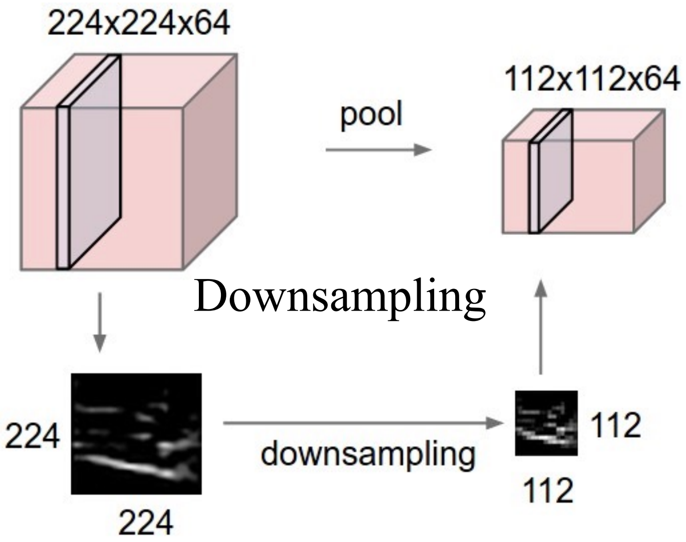
What if without paddings on the border?

- *The spatial dimensions of the input will be changed, causing information loss on the border*

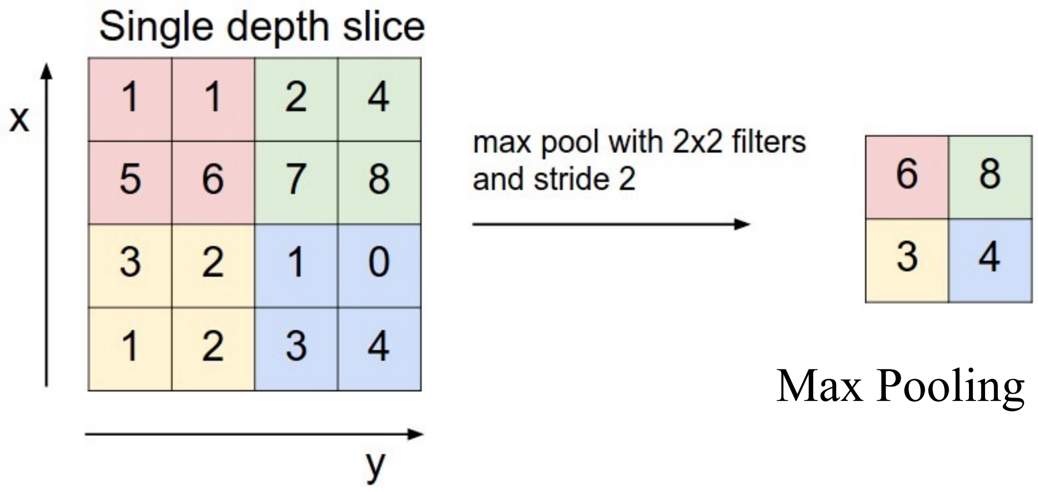
Pooling Layer

Downsampling the spatial dimensions of the input volume

- A network-wise regularization
  - Progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network, and hence to also control overfitting
  - Operates over each activation map independently
  - Usually, no need to zero padding (no convolutional operations)



Downsampling



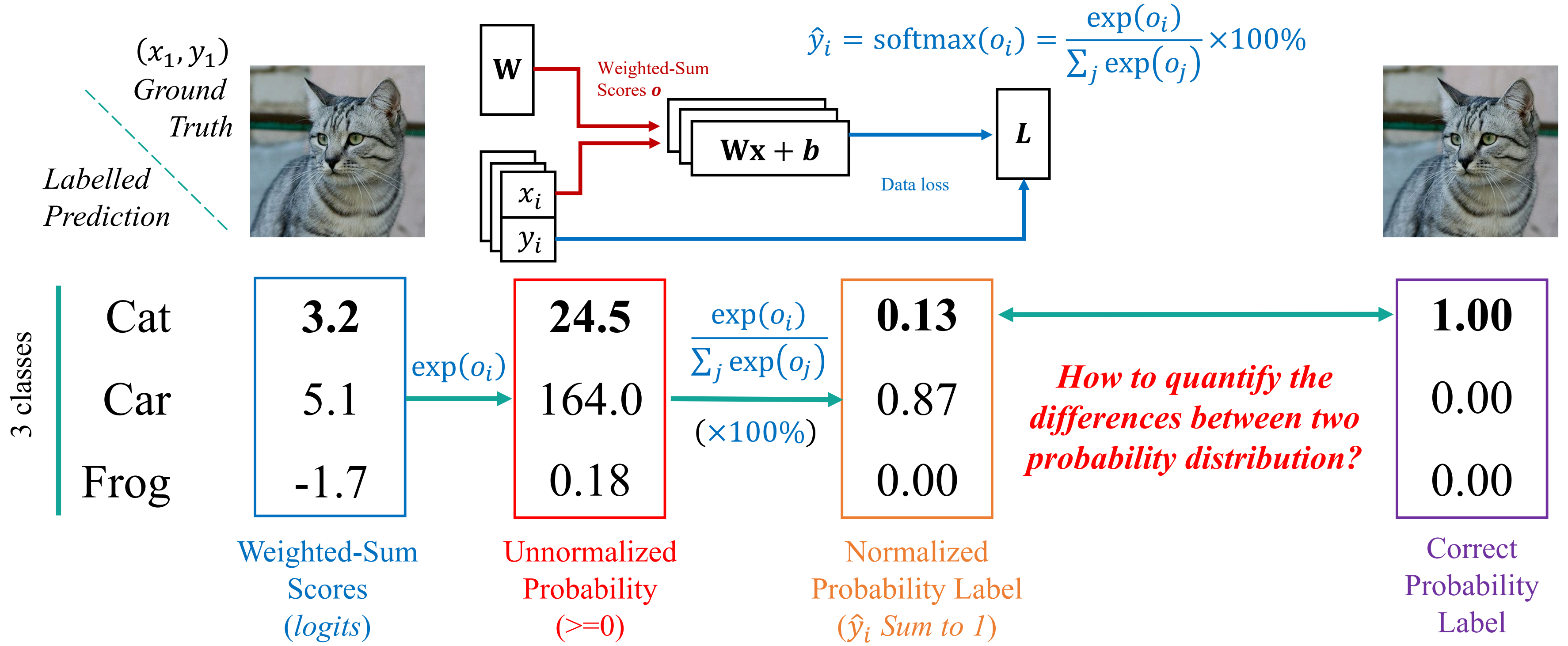
Max Pooling

AncoraSIR.com

Fully-Connected Layer

Full connections to all activations in the previous layer, as seen in regular Neural Networks

- Contains neurons that connect to the entire input volume
- Softmax is a common choice



AncoraSIR.com

ConvNet Architectures

Common choice of hyperparameters of ConvNet designs

- $\text{INPUT} \rightarrow \left[[\text{CONV} \rightarrow \text{RELU}] * \text{N} \rightarrow \text{POOL?}\right] * \text{M} \rightarrow [\text{FC} - \text{RELU}] * \text{K} \rightarrow \text{FC}$
  - the * indicates repetition,
  - the POOL? indicates an optional pooling layer.
  - N >= 0 (and usually N <= 3), M >= 0, K >= 0 (and usually K < 3)

- **INPUT** (that contains the image) should be divisible by 2 many times
  - 32 (e.g. CIFAR-10), 64, 96 (e.g. STL-10), or 224 (e.g. ImageNet), 384, and 512
- **CONV** should be using small filters using a stride of S=1
  - 3x3 or at most 5x5 with zero padding of the input volume
- **POOL** downsamples the spatial dimensions of the input
  - Common setting is to use max-pooling with 2x2 receptive fields with a stride of 2

AncoraSIR.com

A Universal Workflow

仿生设计与学习实验室
Bionic Design & Learning Lab
AncoraSIR.com

SUSTech
Southern University of Science and Technology

Defining the Problem

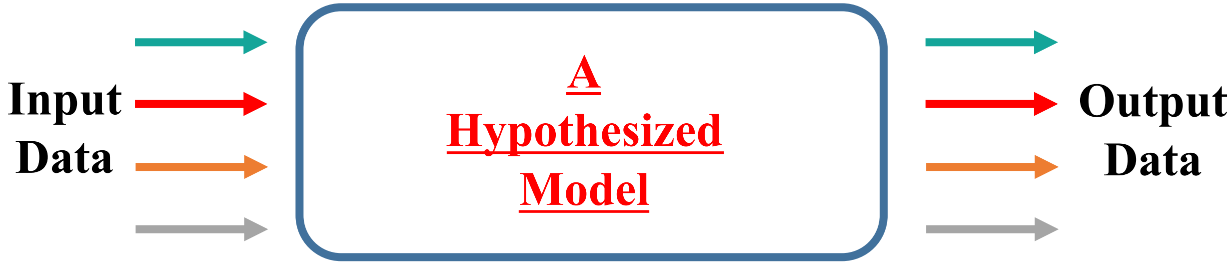
Assembling a Dataset

- **Data availability**
  - Usually the limiting factor at this stage.
- **Identifying the problem type**
  - Guide your choice of model architecture, loss function, etc.
- **A Hypothesis of Modeling**
  - *Can you build a model using price history to predict stock market? ☹*
- **Non-stationary Problem?**
  - *Will your data change over time?*

AncoraSIR.com

What will your input data be?
What are you trying to predict?

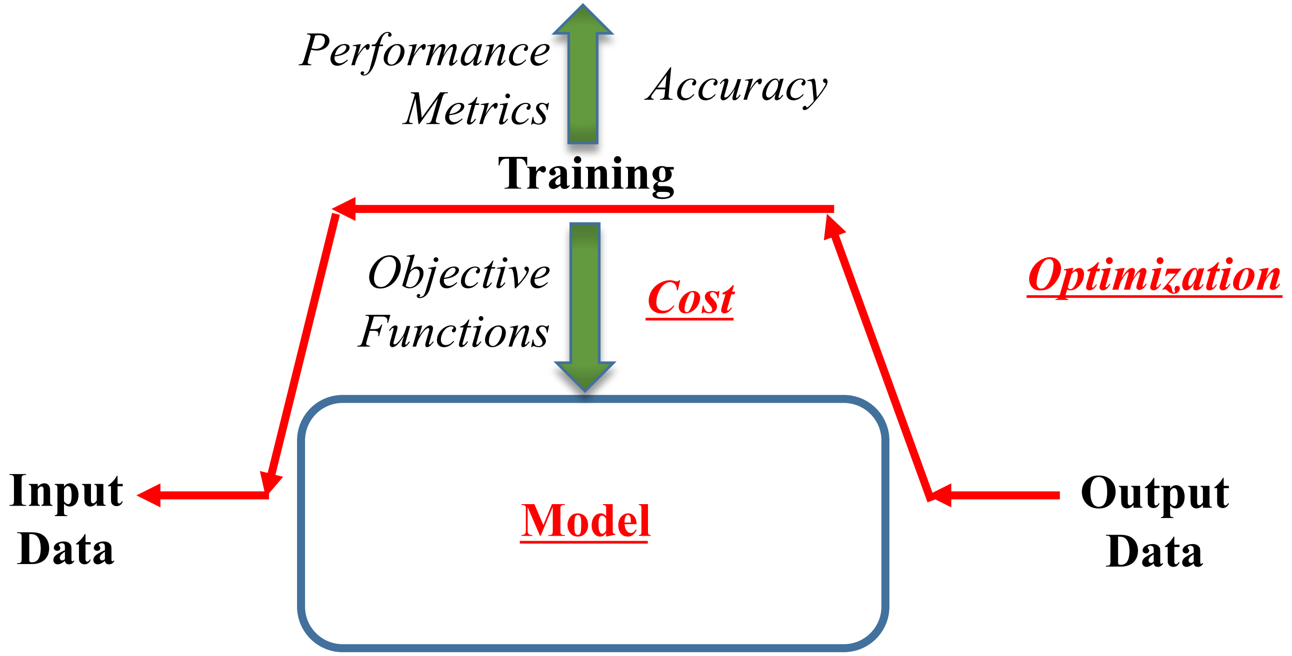
- Can only learn to predict something **if you have available training data**
- Can only be used to **learn patterns** that are present in training data
  - *A fundamental assumption that the future will behave like the past*



- *What type of problem are you facing?*
- *Is it binary classification?*
- *Multiclass classification?*
- *Scalar regression?*
- *Vector regression?*
- *Multiclass, multilabel classification?*
- *Something else, like clustering, generation, or reinforcement learning?*

Choosing a Measure of Success

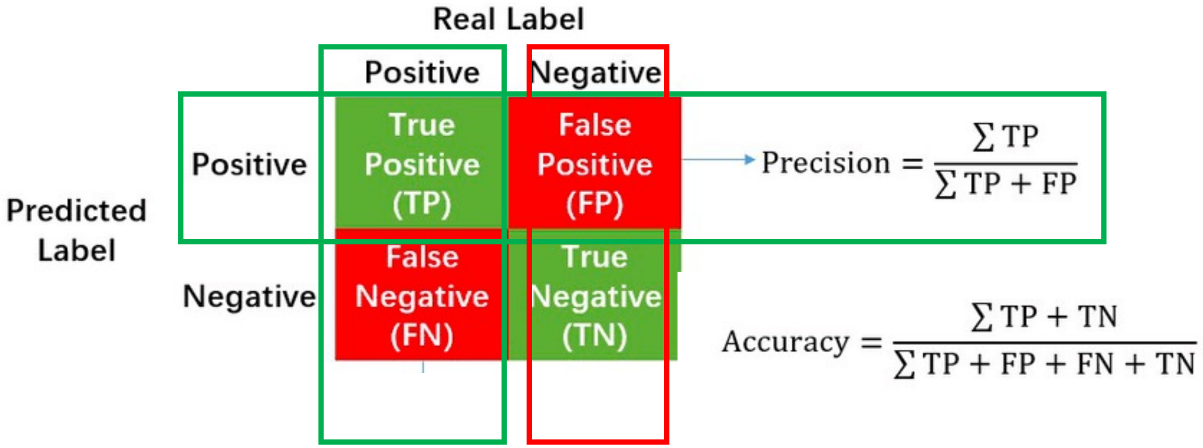
You need to be able to observe it to control it



AncoraSIR.com

Choosing
a Measure of Success

You need to be able to observe it to control it



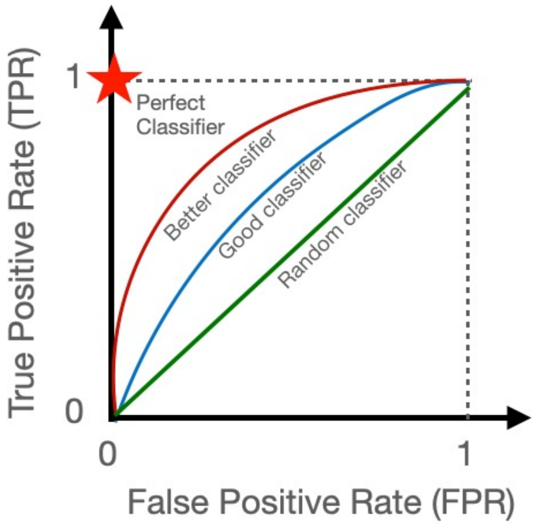
Your metric for success will guide the choice of a loss function

- Accuracy? Precision and recall? **What your model optimizes?**
- For balanced-classification problems (every class is equally likely): *accuracy and area under the receiver operating characteristic curve (ROC AUC)* are common metrics
- For class-imbalanced problems, you can use precision and recall

True Positive Rate (TPR)
= TP / (TP + FN)
= efficiency to identify the signal (Recall)

False Positive Rate (FPR)
= FP / (FP + TN)
= inefficiency to reject background

TP=63	FN=37	100
FP=28	TN=72	100
91	109	200



AncoraSIR.com

Source

Choosing
a Measure of Success

You need to be able to observe it to control it

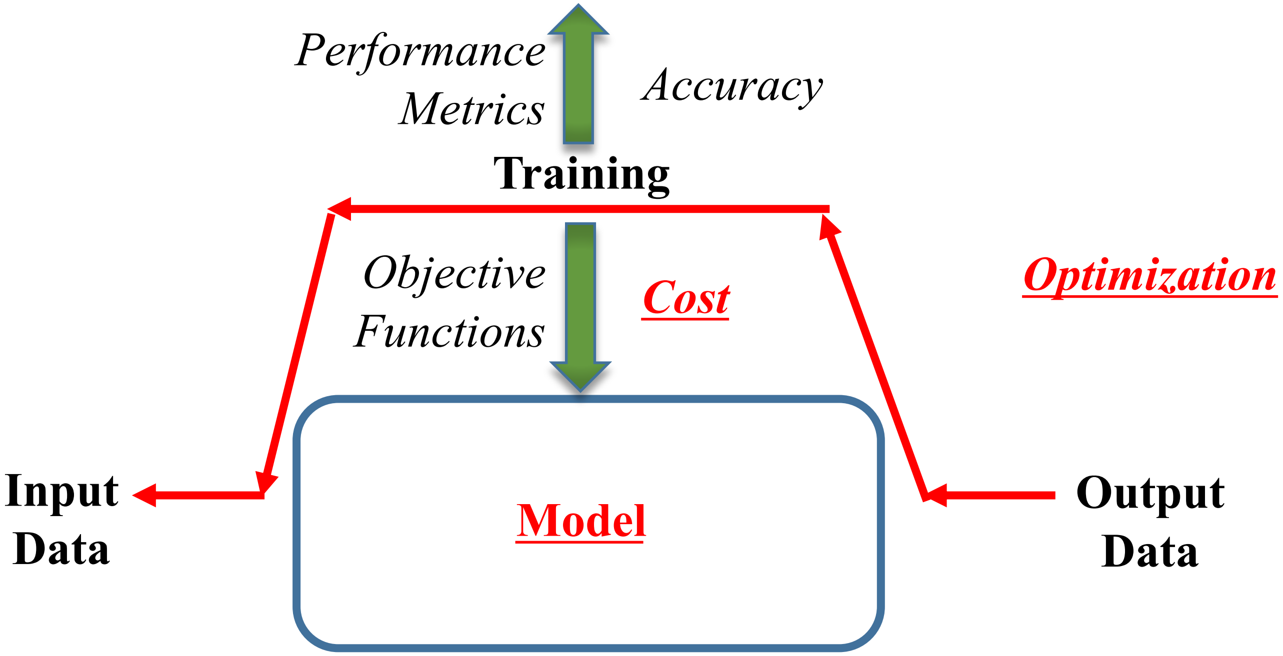
Your metric for success will guide the choice of a loss function

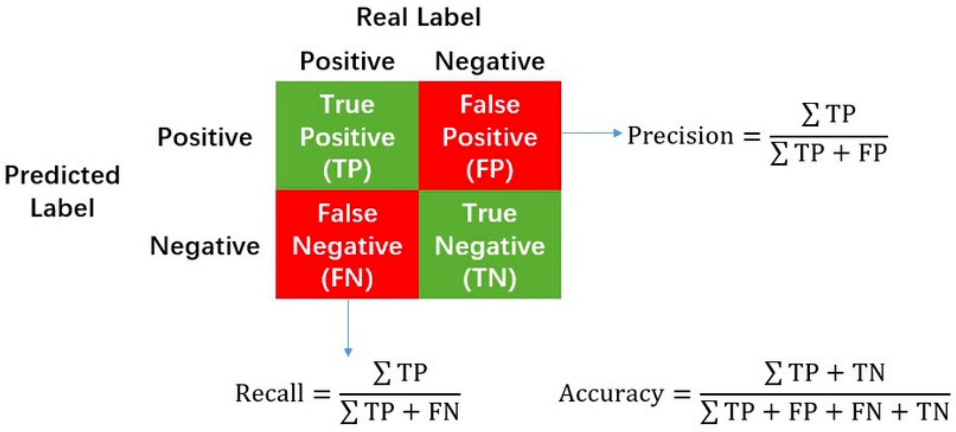
- Accuracy? Precision and recall? **What your model optimizes?**
- For balanced-classification problems (every class is equally likely): *accuracy and area under the receiver operating characteristic curve (ROC AUC)* are common metrics
- For class-imbalanced problems, you can use precision and recall

It isn't uncommon to have to **define your own custom metric**.

- Common in Robotics
- Read more papers
- Be clear and direct about your modeling goal

AncoraSIR.com





Deciding on an Evaluation Protocol

How you'll measure your current progress

Setting Hyperparameters

Idea #1: Choose hyperparameters that work best on the data

BAD: K = 1 always works perfectly on training data

Your Dataset

Maintaining a hold-out validation set

- The way to go when you have plenty of data

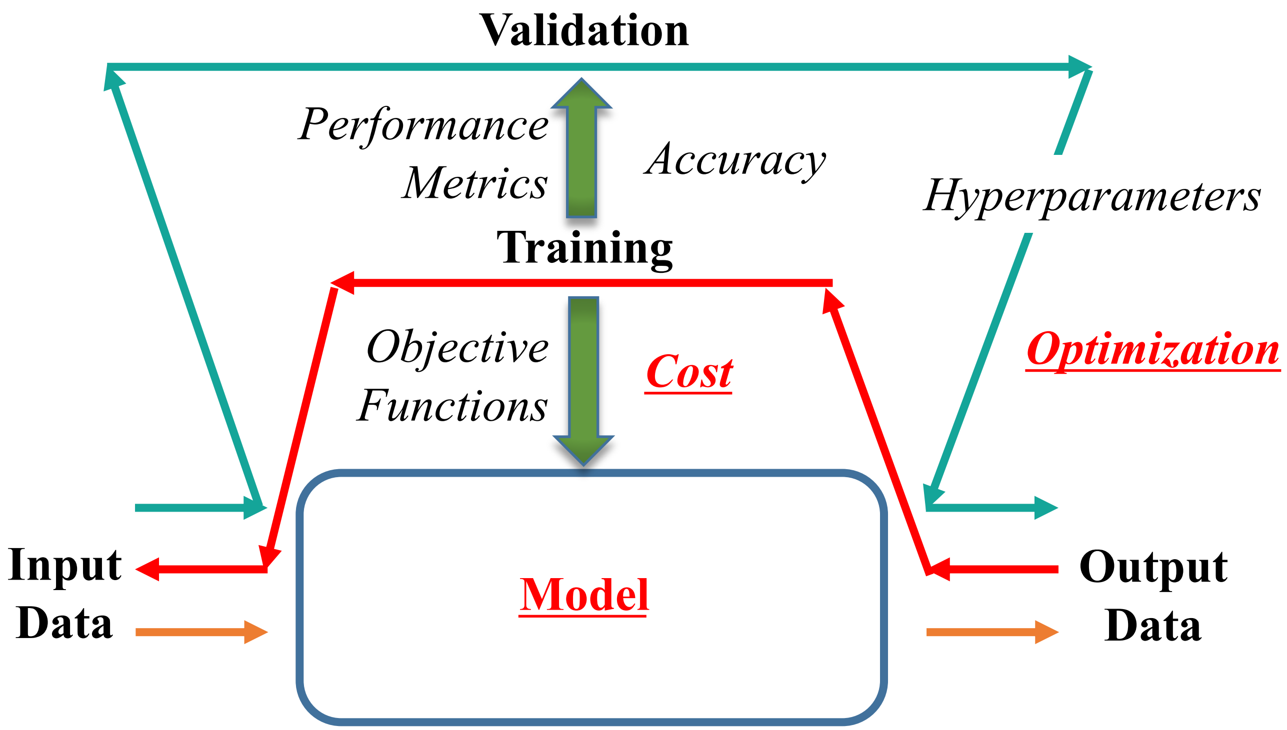
Doing K-fold cross-validation

- The right choice when you have too few samples for hold-out validation to be reliable

Doing iterated K-fold validation

- For performing highly accurate model evaluation when little data is available

AncoraSIR.com



Idea #2: Split data into **train** and **test**, choose hyperparameters that work best on test data

BAD: No idea how algorithm will perform on new data

train	test

Idea #3: Split data into **train**, **val**, and **test**; choose hyperparameters on val and evaluate on test

Better!

train	validation	test

Idea #4: **Cross-Validation**: Split data into **folds**, try each fold as validation and average the results

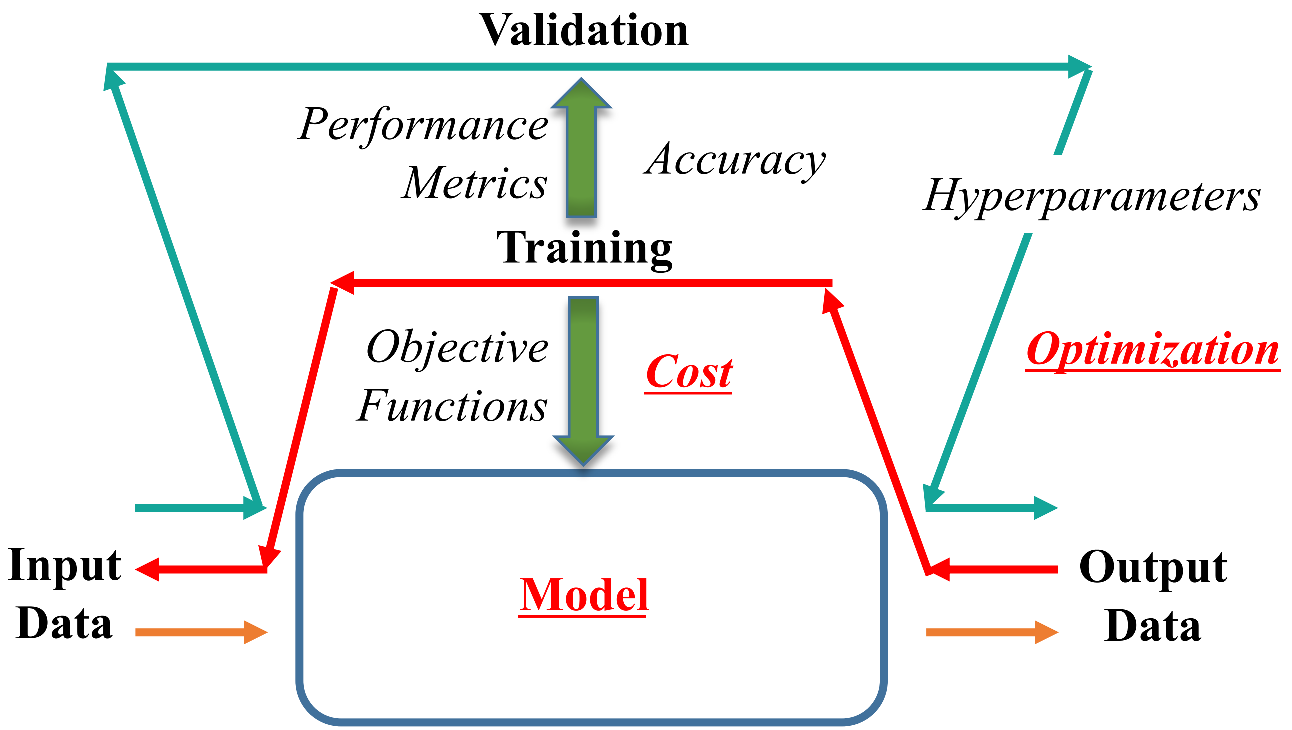
fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test
fold 1	fold 2	fold 3	fold 4	fold 5	test

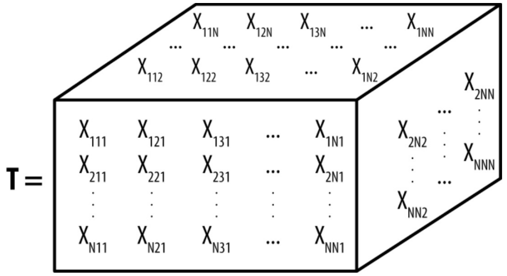
Useful for small datasets, but not used too frequently in deep learning

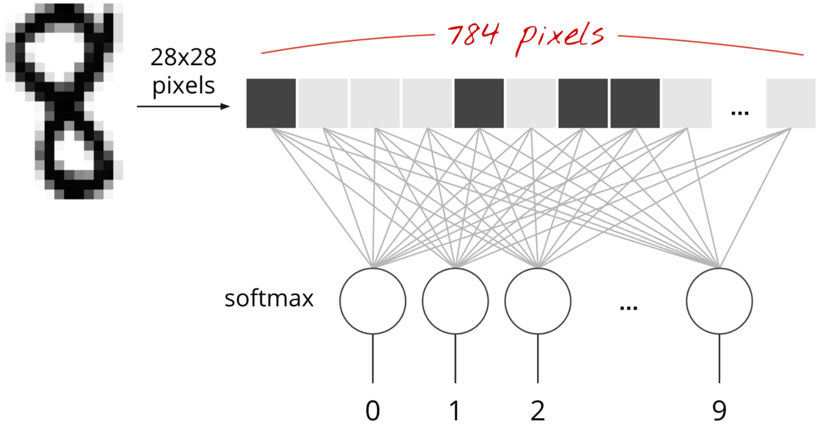
Preparing Your Data

Format your data in a way that can be fed into a model of deep neural network

- Your data should be formatted as **tensors**
- The values taken by these tensors should usually be **scaled** to small values
  - For example, in the [-1, 1] range or [0, 1] range
- **Normalize** the data if different features take values in different ranges





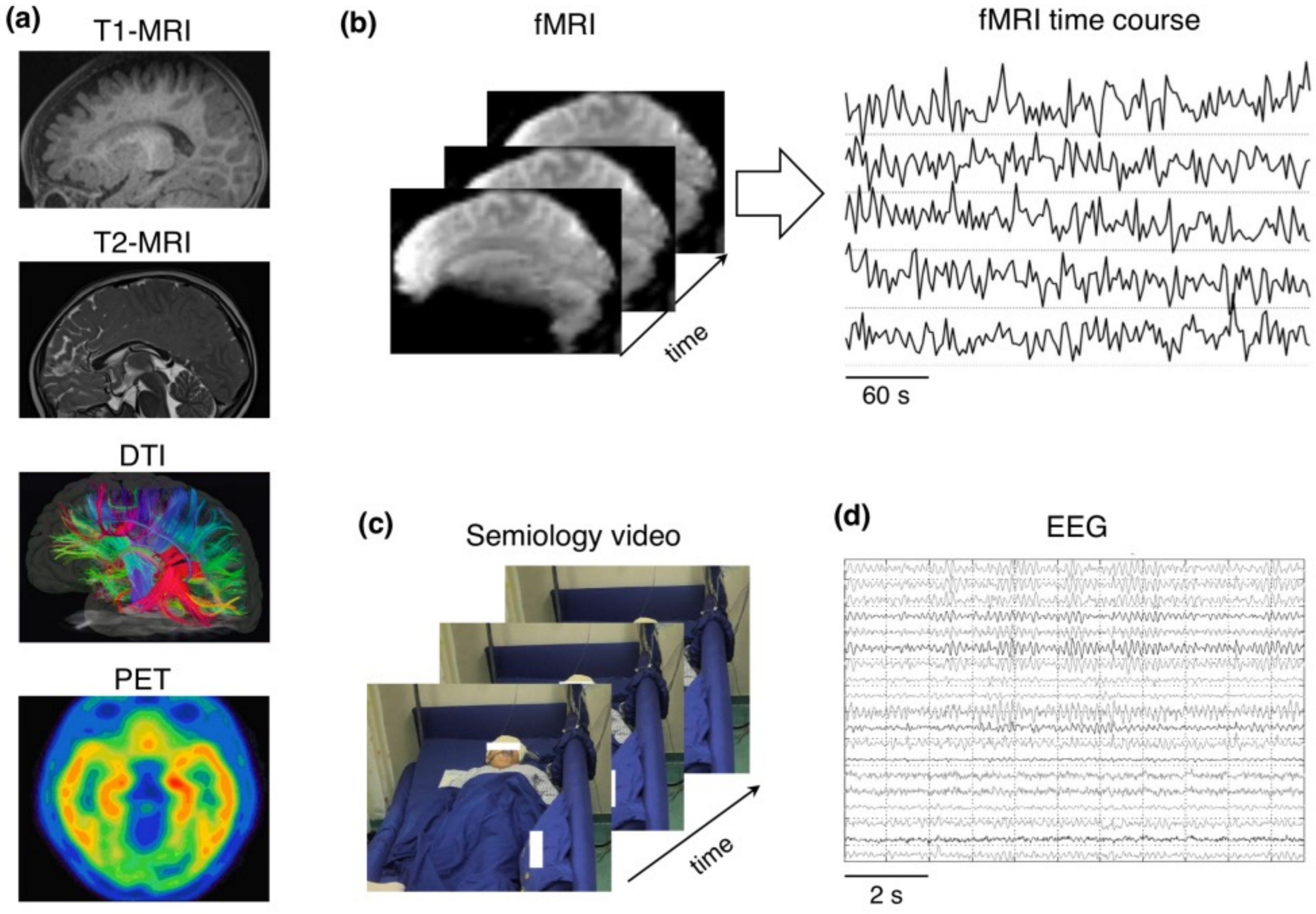


AncoraSIR.com

Preparing Your Data

Format your data in a way that can be fed into a model of deep neural network

- Your data should be formatted as **tensors**
- The values taken by these tensors should usually be **scaled** to small values
    - For example, in the [-1, 1] range or [0, 1] range
- **Normalize** the data if different features take values in different ranges
- **Feature engineering** may be required, especially for small-data problems

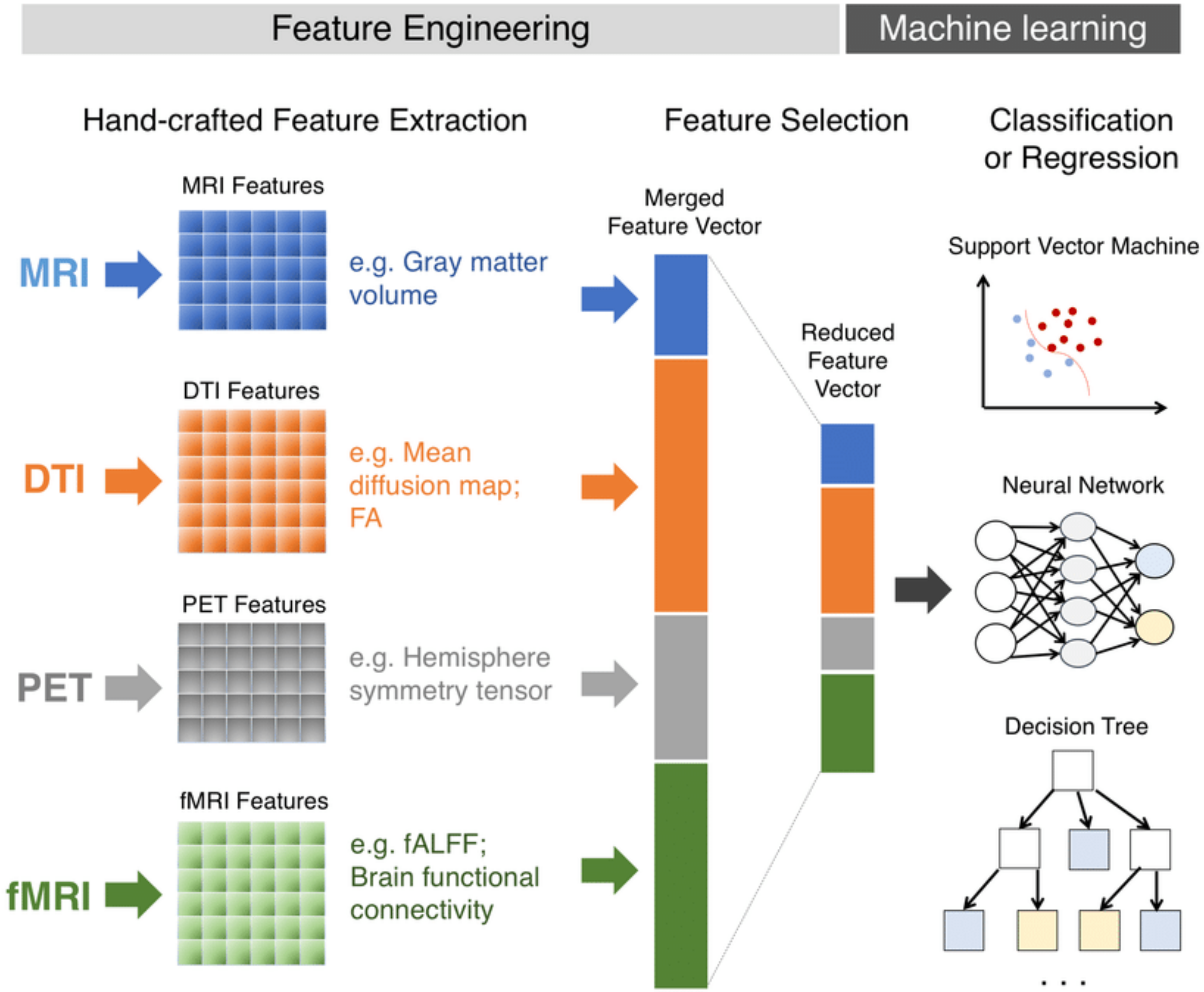


Source

Preparing Your Data

Format your data in a way that can be fed into a model of deep neural network

- Your data should be formatted as **tensors**
- The values taken by these tensors should usually be **scaled** to small values
  - For example, in the [-1, 1] range or [0, 1] range
- **Normalize** the data if different features take values in different ranges
- **Feature engineering** may be required, especially for small-data problems



Source

AncoraSIR.com

Developing a Model Better than a Baseline

To achieve statistical power by building a small model that beats a dumb baseline

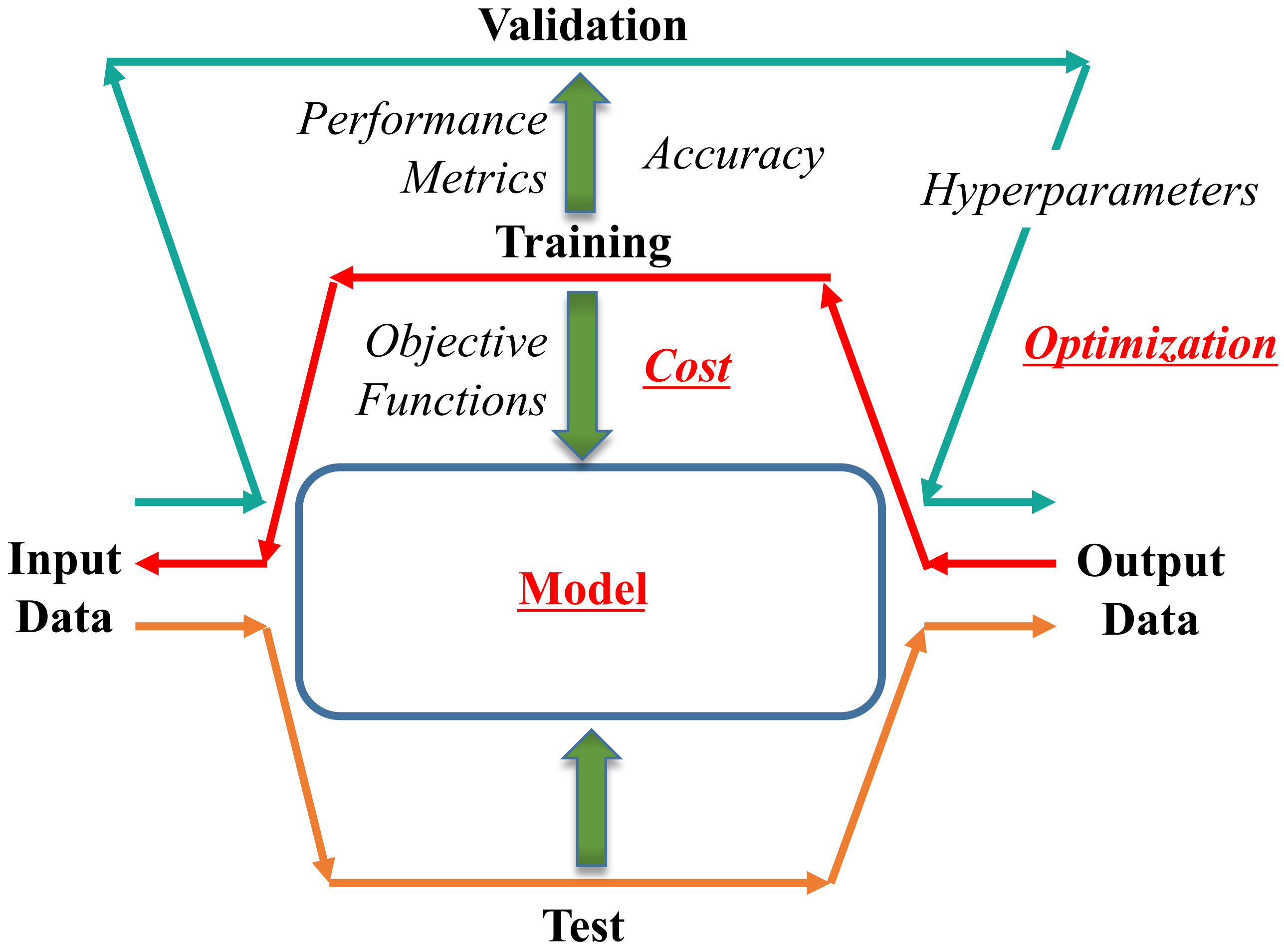
Check your assumptions agains a baseline model

- Your outputs can be predicted given your inputs.
- The available data is sufficiently informative to learn the relationship between inputs and outputs.
- (*asking the right questions to your data*)

Key choices to build a model

- *Last-layer activation*—This establishes useful constraints on the network's output.
- *Loss function*—This should match the type of problem you're trying to solve.
- *Optimization configuration*—What optimizer will you use? What will its learning rate be?

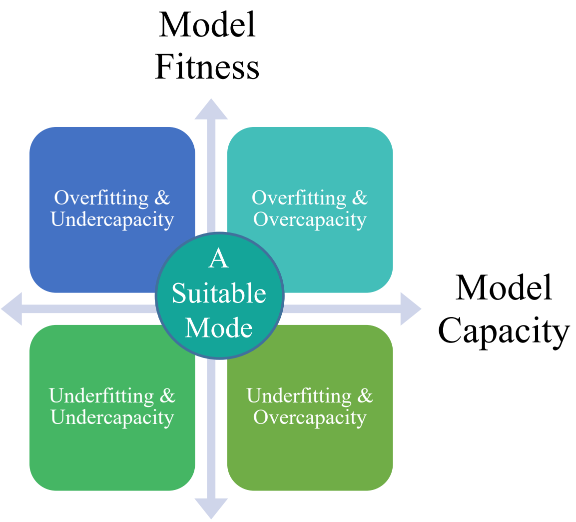
AncoraSIR.com

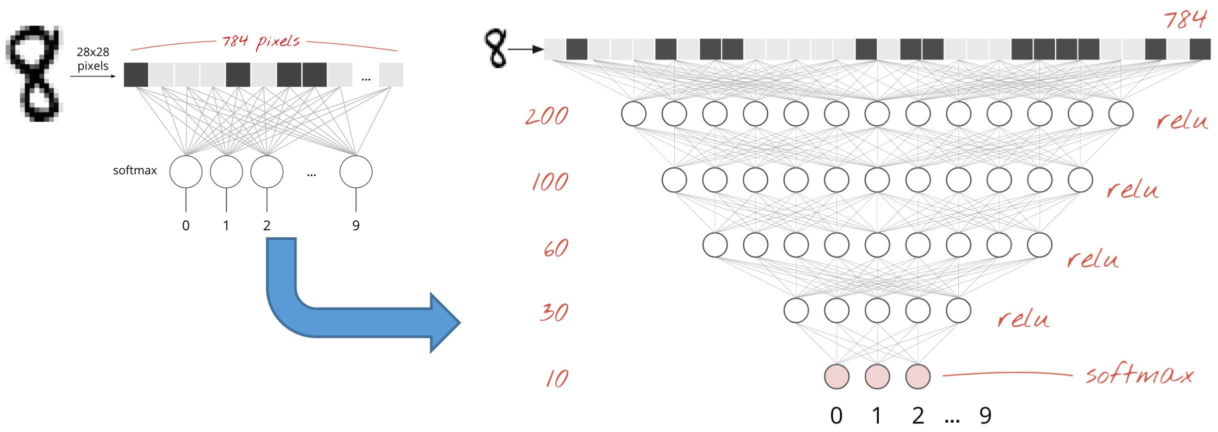


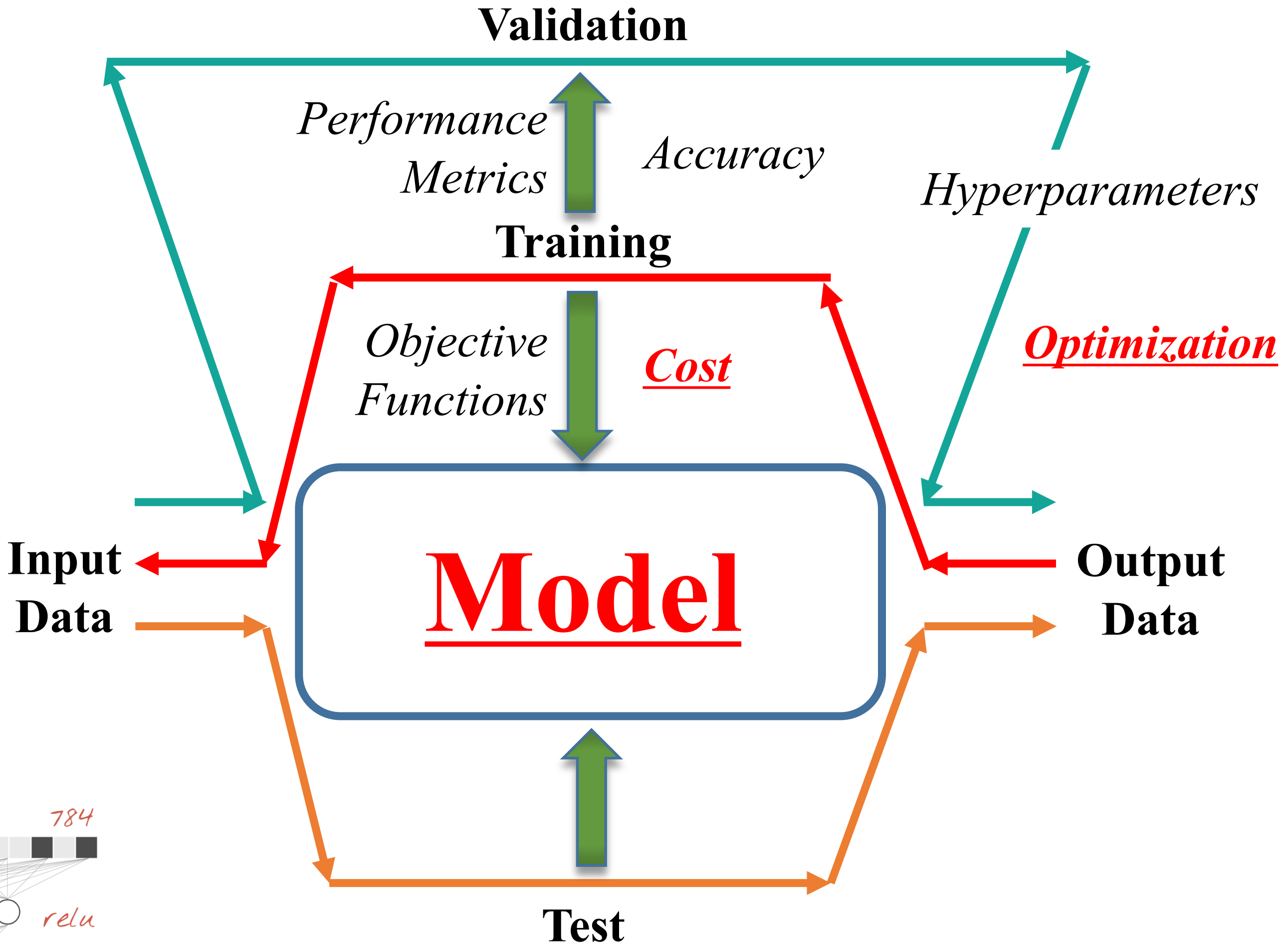
Problem type	Last-layer activation	Loss function
Binary classification	sigmoid	binary_crossentropy
Multiclass, single-label classification	softmax	categorical_crossentropy
Multiclass, multilabel classification	sigmoid	binary_crossentropy
Regression to arbitrary values	None	mse
Regression to values between 0 and 1	sigmoid	mse or binary_crossentropy

Scaling Up: Developing a Model that Overfits

1. Add layers.
2. Make the layers bigger.
3. Train for more epochs.







- *Is your model sufficiently powerful?*
- *Does it have enough layers and parameters to properly model the problem at hand?*

AncoraSIR.com

Regularizing Your Model

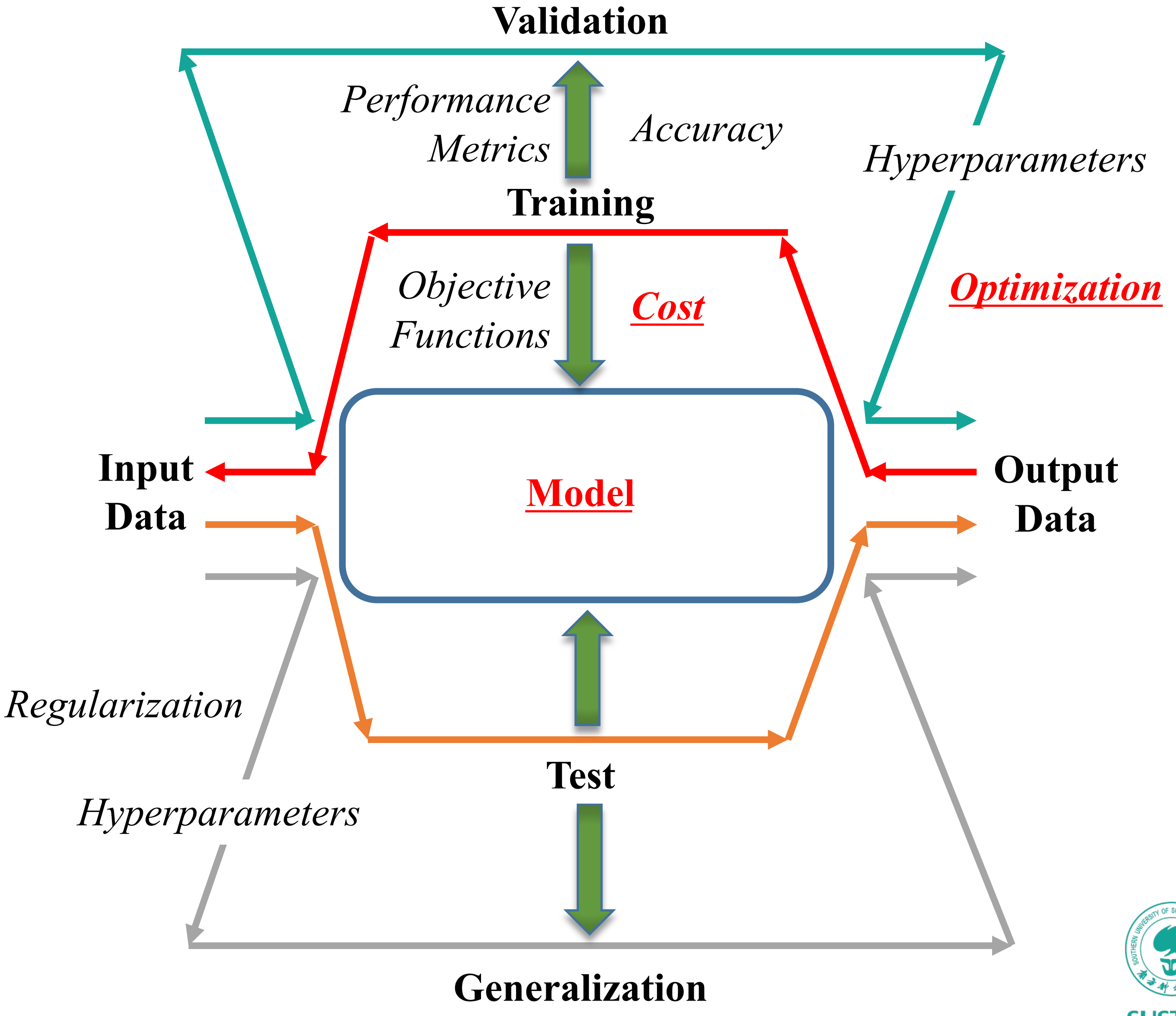
The most time-consuming step …

- *Modify your model, train it, evaluate on your validation data (not the test data, at this point),*
- *Modify it again, and repeat, until the model is as good as it can get*

Things you can try

- Add dropout
- Try different architectures: add or remove layers
- Add L1 and/or L2 regularization
- Try different hyperparameters
  - such as the number of units per layer, or
  - the learning rate of the optimizer
- Optionally, iterate on feature engineering
  - add new features, or
  - remove features that don't seem to be informative

AncoraSIR.com



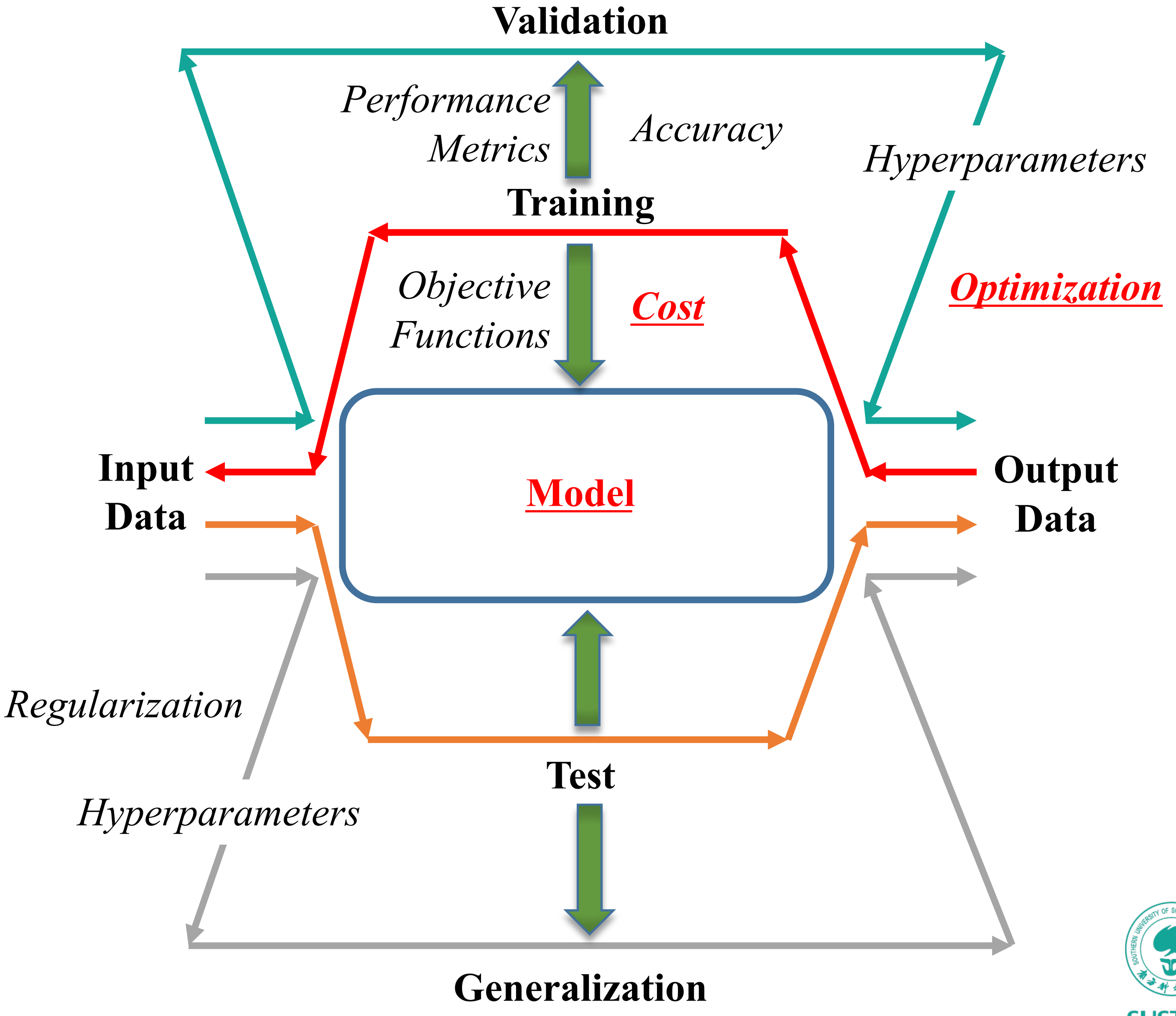
Tuning
Your Hyperparameters

Information Leak => Model Overfit

- Every time you use feedback from your validation process to tune your model, you leak information about the validation process into the model.
- This makes the evaluation process less reliable.

Once you've developed a satisfactory model configuration,

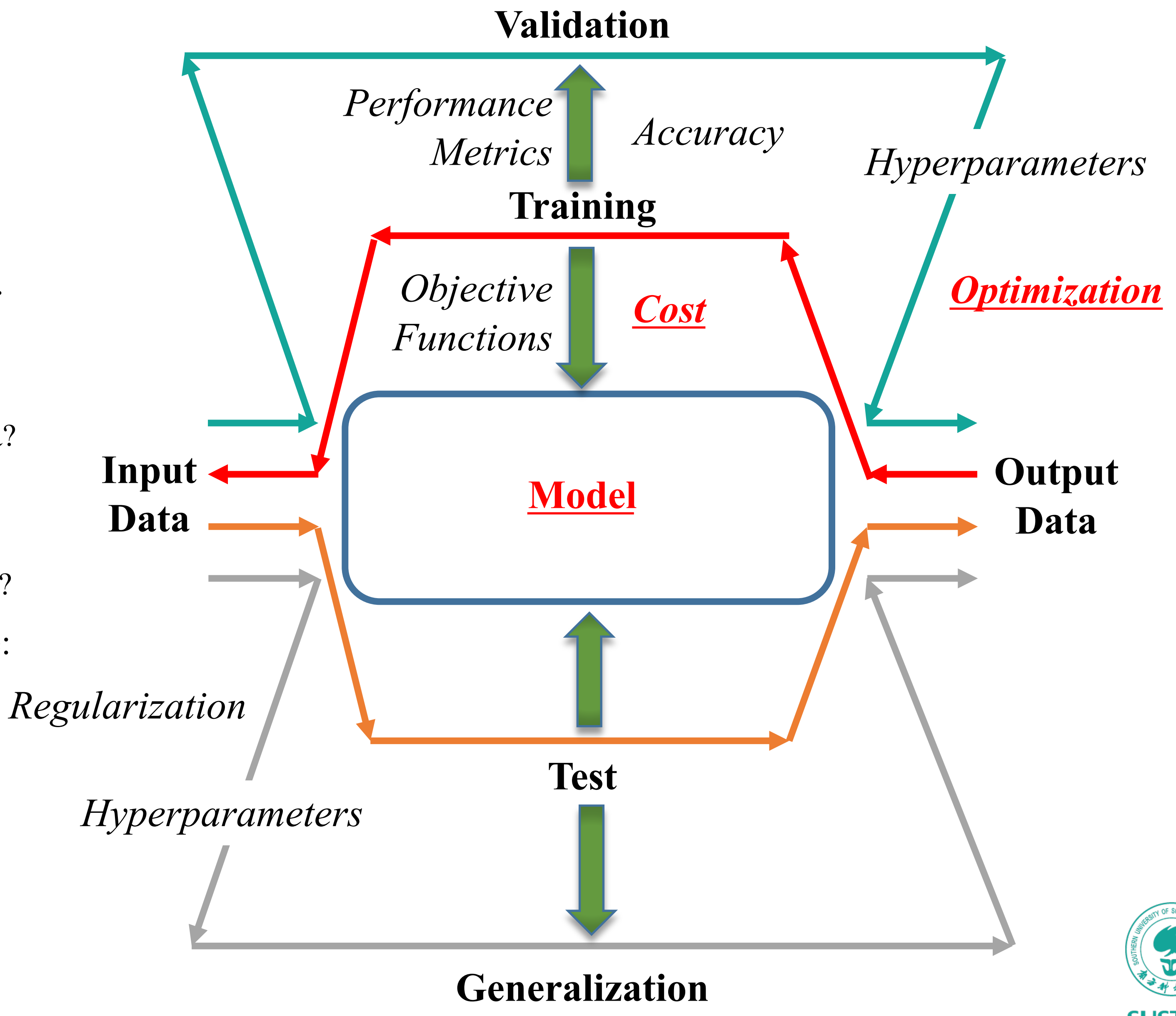
- you can train your final production model on all the available data (training and validation) and
- evaluate it one last time on the test set
- If good, then proceed
- Or if worse, you may want to switch to a more reliable evaluation protocol
(may be caused by overfitting)



AncoraSIR.com

A Universal Workflow

- Define the problem and the data on which you'll train.
  - Collect this data or annotate it with labels if need be.
- Choose how you'll measure success on your problem.
  - Which metrics will you monitor on your validation data?
- Determine your evaluation protocol:
  - Hold-out validation? K-fold validation?
  - Which portion of the data should you use for validation?
- Develop a model that does better than a basic baseline:
  - A model with statistical power.
- Develop a model that overfits.
- Regularize your model and tune its hyperparameters,
  - Based on performance on the validation data.
  - A lot of machine-learning research tends to focus only on this step—but keep the big picture in mind.



Learning a Transition Model

Learning a Transition Model of the Environment

The robot must learn a model of how its actions affect the task state, and the resulting background cost, for use in planning

Transition Models of the Environment

Continuous, Discrete and Hybrid Models

↓

Stochasticity ← **Probabilistic Models** → **Uncertainty**

↓

Learning a New Model

Self-supervision and Exploration

Reuse an Existing Model

Transferring and Reusing

AncoraSIR.com

SUSTech Southern University of Science and Technology

Continuous Transition Models

*Regression methods can be used to learn **low-level** transition models for predicting the next state as a set of continuous values, even when the set of actions is discrete*

Scoop & Dump Task $l(h_0, a_0, ..., a_T, h_g) = \|\mathcal{F}(h_0, a_0, ..., a_T) - h_g\|_1$

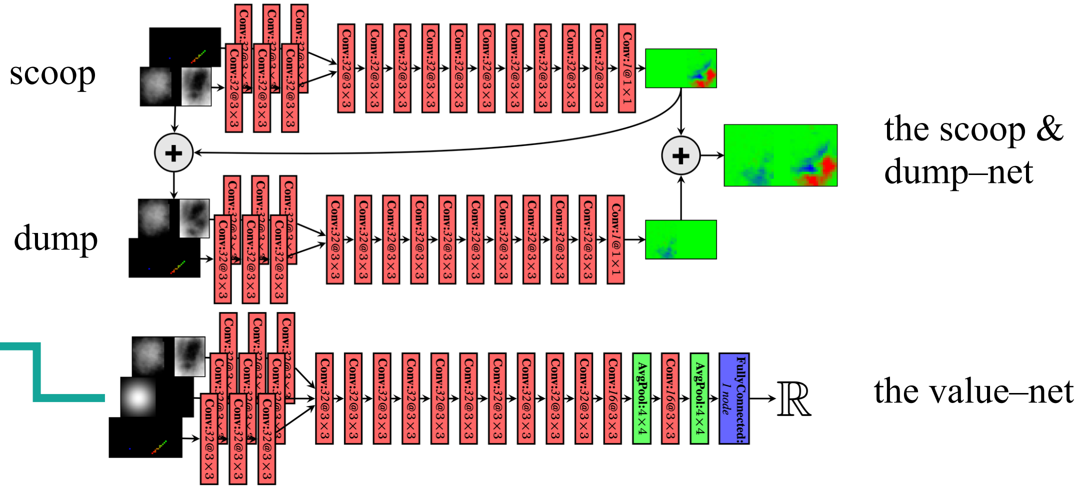
h_0: a given ***initial state*** of the environment

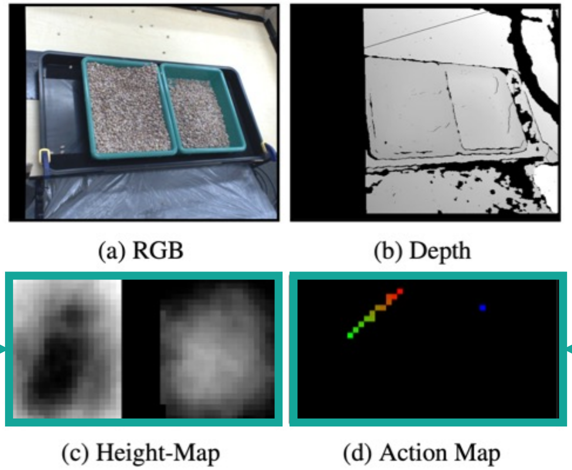
h_g: a given ***goal state*** of the environment

$a_{0,...,T}$: a series of ***robot actions***

$\|Norm\|_1$: L1 norm to be minimized as distance to the goal state

$\mathcal{F}(h_0, a_{0,...,T}) = h_{T+1}$ applies actions sequentiall to reach h_{T+1}



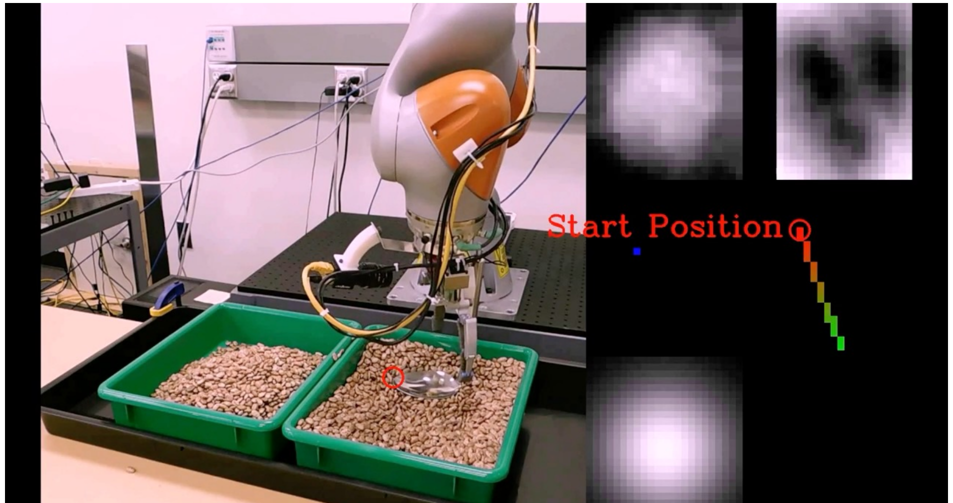


(a) RGB (b) Depth

(c) Height-Map (d) Action Map

Robot action (a 9D vector)

- *Scoop action*
  - the start location (2D)
  - the start angle (1D)
  - the end location (2D)
  - the end angle (1D)
  - the roll angle (1D)
- *Dump action*
  - the dump location (2D)



https://arxiv.org/pdf/1709.02833.pdf

AncoraSIR.com

Thank you~

Wan Fang
Southern University of Science and Technology