b
UNIVERSITÄT
BERN

Deep Feedforward Networks - Regularization

Simon Jenni
(slides by Paolo Favaro)

Contents

- Regularization in Feedforward Neural Networks
  - Parameters, optimization, dataset augmentation, I/O noise, semi-supervised learning, early stopping

- Based on **Chapter 7** of Deep Learning by Goodfellow, Bengio, Courville

Regularization

- A central problem in ML is **generalization**: How do we design an algorithm that can perform well not only on training data but also on new data?

- Regularization aims at reducing the generalization error of an algorithm

Generalization

- Problems with generalization (see also Machine Learning Review slides)
  - Underfitting (large bias but low variance)
  - Overfitting (small bias but high variance)
- Neural networks typically are in the second case and regularization aims at reducing variance

Regularization

- Strategies
  - Constrain model (e.g., restrict model family or parameter space)
  - Add terms to loss function (equivalent to soft constraints to the model) — can encode priors
  - Ensemble methods (combine multiple hypotheses)

Dataset Augmentation

- The best way to make the model generalize well is to train it on more data
- One way to augment our dataset is to apply a number of realistic transformations to the data we already have and create new synthetic samples, which share the same label
- This process of data manipulation is also called **jittering**

Dataset Augmentation

original

affine distortion

horizontal flip

noise

random translation

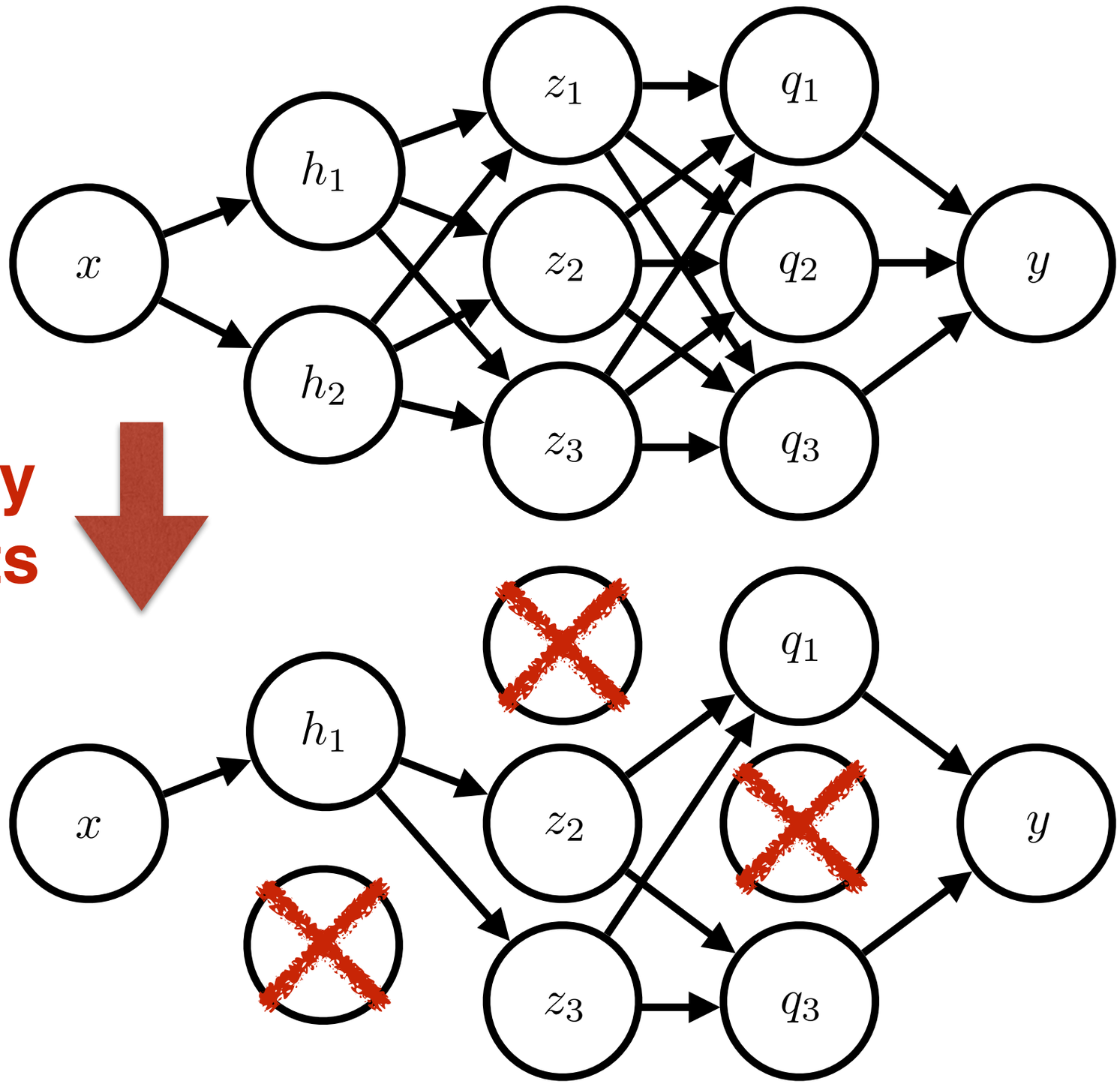
elastic deformation

hue shift

Noise Robustness

- Apply noise to the input data at each iteration
- Apply noise to the inputs of the hidden units (Poole et al 2014)
- **Dropout** can be seen as multiplicative noise

Dropout



Randomly drop units

Noise Robustness

- Apply noise to the weights
  - Model weights as random variables
  - Encourage stability of learned mapping (weights find minima with a flat neighborhood)

Label Smoothing

- Labels might be wrong (remember: it is human annotation)
- Let us model noise in the labels
  - For example, $p(y) = (1-\epsilon)\hat{p}(y) + \epsilon\mathcal{U}[1, K]$

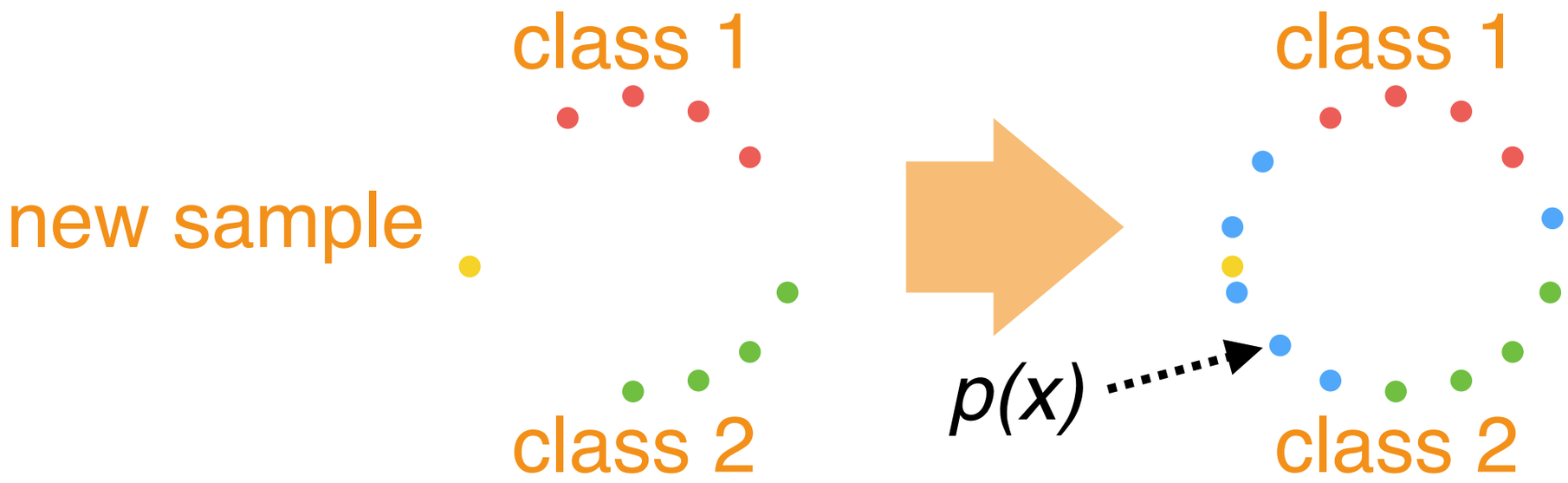
    labeling
  - **Label smoothing** replaces 0s and 1s with

    $\frac{\epsilon}{K-1}$ and $1-\epsilon$ respectively

Semi-Supervised Learning

- Semi-supervised learning uses unlabeled samples from $p(x)$ and labeled samples from $p(x,y)$ to build $p(y|x)$ or directly predict y from x

- The probability density $p(x)$ can be seen as a prior on the input data



Early Stopping

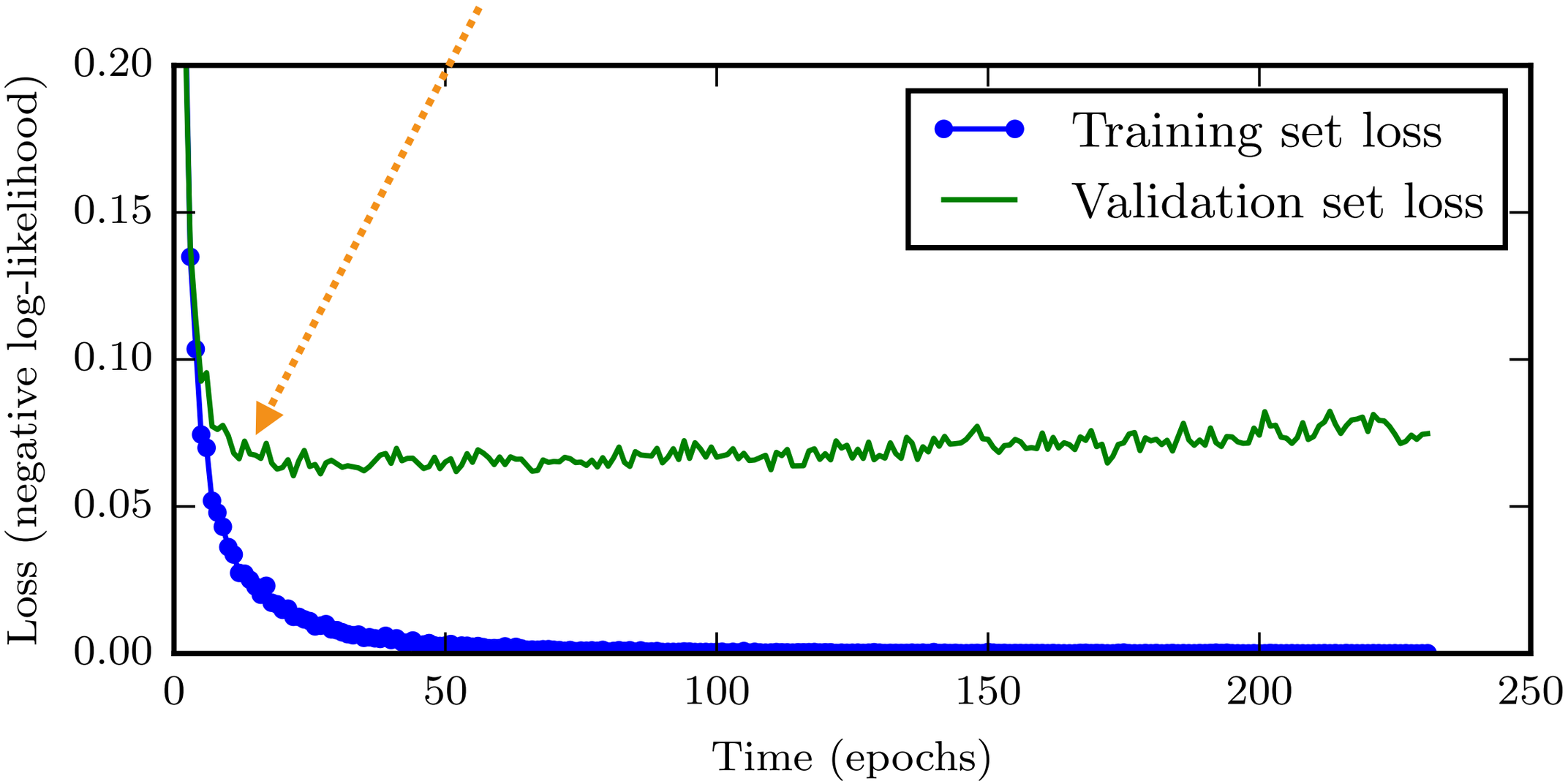
- Neural networks require iterative algorithms for training (typically a gradient descent-type)

- The larger the number of iterations and the lower the training error

- A technique to increase the generalization of the model is to **limit the number of iterations**

Early Stopping

terminate while validation set performance is better



Early Stopping

- Since the validation set is not used for training, after early stopping one can either

  1) retrain the network on all the data (training + validation sets) and then stop after the same number of steps of the early stopping or

  2) continue training the network on all the data (training + validation sets) and then stop when the loss on the validation set is below the loss on the training set (at the early stopping iteration time)

b
UNIVERSITÄT
BERN

Deep Feedforward Networks - Optimization

Paolo Favaro

Contents

- Optimization in Feedforward Neural Networks
    - Batch and mini batch algorithms, stochastic gradient descent, weight initialization

- Based on **Chapter 8** of Deep Learning by Goodfellow, Bengio, Courville

Batch and Minibatch Algorithms

- **Batch** or **deterministic** gradient methods use the whole training set at each iteration

- **Minibatch stochastic** gradient methods use a batch of samples at each iteration

- **Stochastic** gradient methods use only one sample at each iteration

- Today, it is common practice to call minibatch stochastic simply stochastic

Choice of the Batch Size

- Larger batches give better gradients, but the estimate improvement is low
- Small batches might underutilize multicore architectures
- Examples in a batch are processed in parallel; amount of memory defines the maximum size
- GPUs may prefer sizes that are a power of 2
- Small batches may have a regularization effect

Choice of the Batch Size

- The size depends also on the gradient method
  - Methods based on only the loss gradient require small batch sizes
  - Methods based on higher order derivatives (e.g., Hessian) require large batch sizes (to compensate for the larger approximation error)

Shuffling

- An unbiased estimate of the expected gradient requires independent samples
- Using data where the order is fixed might lead to batches where all samples are highly correlated
- Common practice is to randomly visit the training set
- Can save a dataset where the data has been randomly permuted (**data shuffling**)

Basic Algorithms

Stochastic Gradient Descent

- Learning rate ϵ_k and initial parameter θ
- **while** (stopping criterion not met) **do**
  - Sample a minibatch of ***m*** examples from the training set with the corresponding targets
  - Compute gradient estimate $\hat{g} \leftarrow \frac{1}{m} \nabla_\theta \sum_i L(f(x_i; \theta), y_i)$
  - Apply update $\theta \leftarrow \theta - \epsilon_k \hat{g}$
- **end while**

Stochastic Gradient Descent

- Probably the most used algorithm in deep learning
- Main setting is the learning rate ϵ_k
- It is necessary to gradually decrease the learning rate over iteration time ***k***
- Sufficient conditions (in addition to others on the cost) to guarantee convergence of SGD are that

$$\sum_{k=1}^{\infty} \epsilon_k = \infty \qquad \sum_{k=1}^{\infty} \epsilon_k^2 < \infty$$

Weight Initialization Strategies

- Since the optimization problem is non convex, initialization determines the quality of the solution

- Current initialization strategies are simple and heuristic

- Some initial points may be beneficial to the optimization task, but not to generalization

- One criterion is that the initial parameters need to **break the symmetry** between different units

Weight Initialization Strategies

- Two hidden units with the same activation function and inputs should have different initial parameters

- Otherwise a deterministic learning algorithm will update both of these units in the same way

- The goal of diversifying the computed functions motivates random initialization

- Random weights can be obtained from a Gaussian or Uniform distribution

Learning Based Initialization Strategies

- Another strategy is to initialize weights by **transferring weights** learned via an unsupervised learning method

- This is also a technique called **fine-tuning** which aims at exploiting small annotated datasets by combining them with large unlabeled ones

Adaptive Learning Rates

- The learning rate is one of the most difficult parameters to set
- It has a significant impact on the model performance
- It is therefore treated as a hyperparameter that requires adjustment during training

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Choosing the Optimization Algorithm

- Currently there is no consensus on what algorithm performs best

- Most popular choices are: SGD, SGD+Momentum, RMSProp, RMSProp+Momentum, AdaDelta, Adam

- Strategy: Pick one and get familiar with the tuning

UNIVERSITÄT
BERN

Convolutional Neural Networks

Simon Jenni
(slides by Paolo Favaro)

Contents

- Convolutional Neural Networks
  - Convolutions (standard, unshared, tiled)
- Based on **Chapter 9** of Deep Learning by Goodfellow, Bengio, Courville

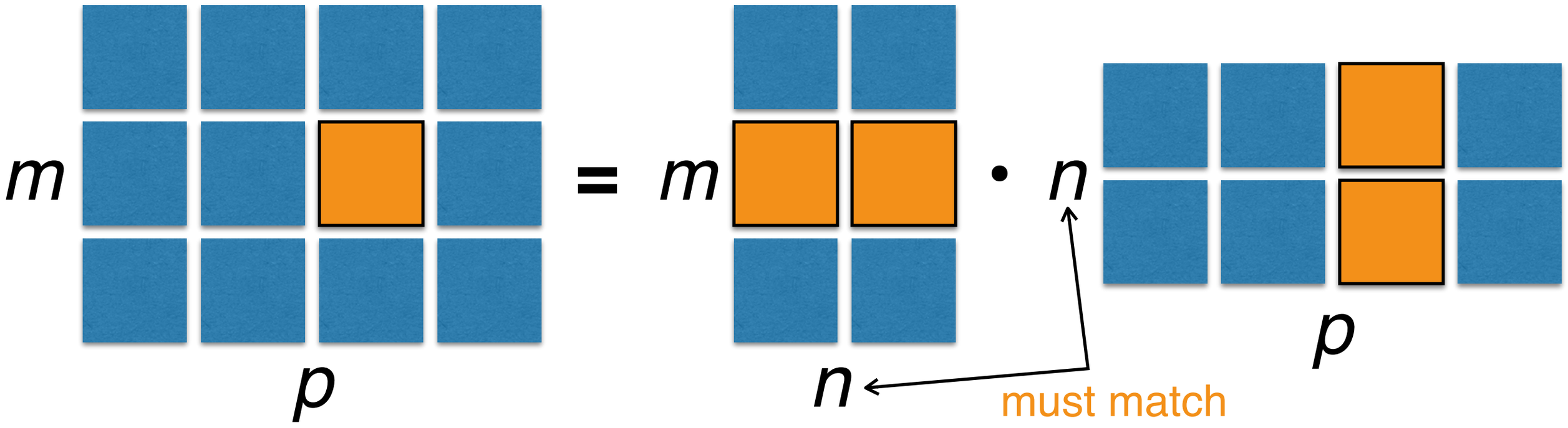
Convolutional Networks

- A specialized neural network for data arranged on a grid (e.g., audio signals, images)

- Allow neural networks to deal with high-dimensional data

- Key idea is to substitute fully connected layers with a convolution

Fully Connected Layers



matrix product

The Convolution Operation

feature map

input

kernel

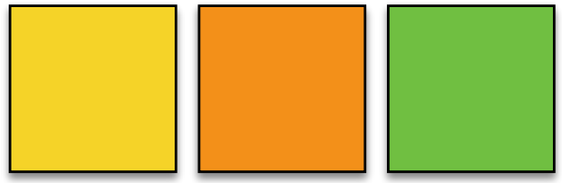
$$s[m,n] = (x * w)[m,n] = \sum_{i,j} x[m-i, n-j] w[i,j]$$

symmetric

$$= \sum_{i,j} w[m-i, n-j] x[i,j]$$

linear in x
with fixed w

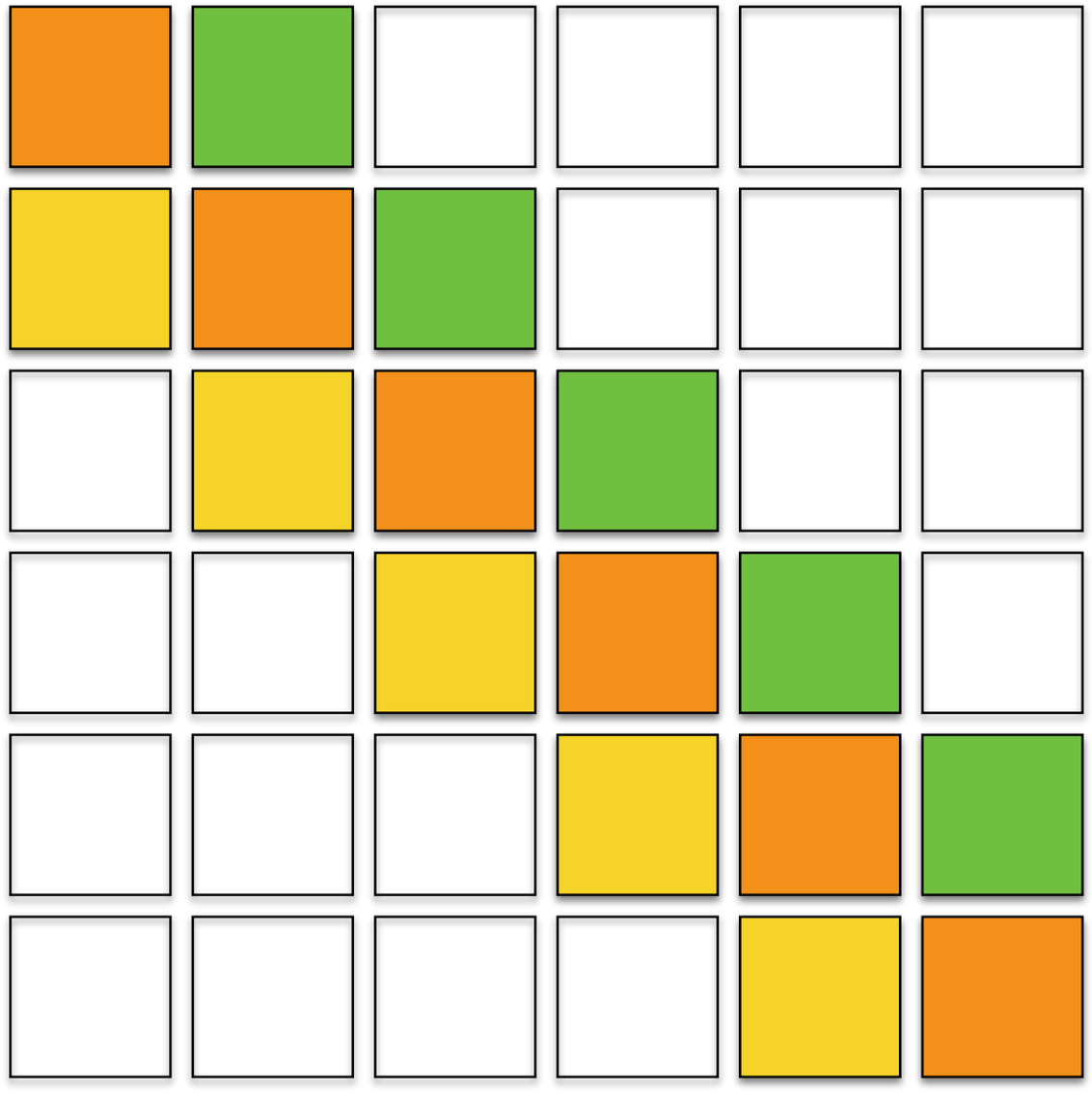
Toeplitz Matrix



kernel

$$s[n] = (x * w)[n]$$

$$= \sum_i A[n, i]x[i]$$



Toeplitz matrix

Variants

- Input data is typically a 4D tensor: 2 dimensions for the spatial domain, 1 dimension for the channels (e.g., colors), and 1 dimension for the batch
- The convolution (correlation) applies to the spatial domain only

$$Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m,k+n} K_{i,l,m,n}$$

output input kernel

Convolution Example

Input

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Kernel

1	0	1
0	1	0
1	0	1

Convolution Operation

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

Output

4		

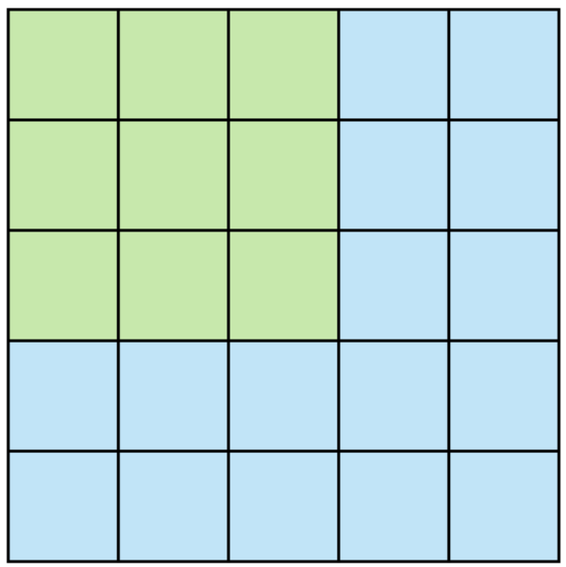
Source: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Stride

- We can also skip outputs by defining a **stride** s larger than 1

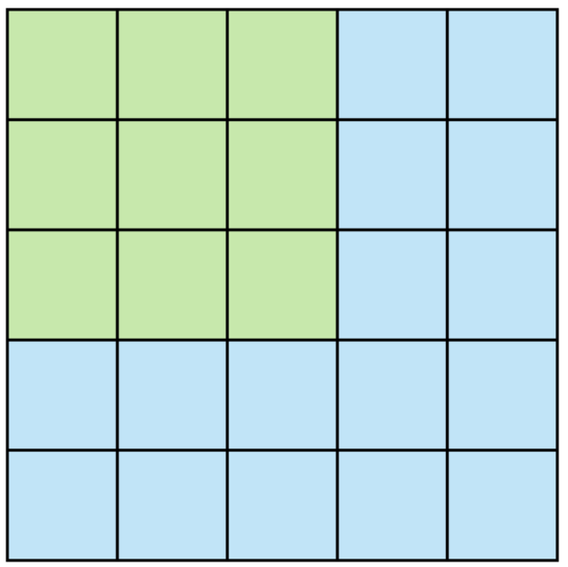
$$Z_{i,j,k} = \sum_{l,m,n} V_{l,j \times s+m,k \times s+n} K_{i,l,m,n}$$

Stride Example

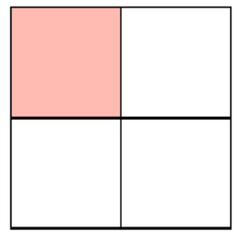


Stride 1

Feature Map



Stride 2



Feature Map

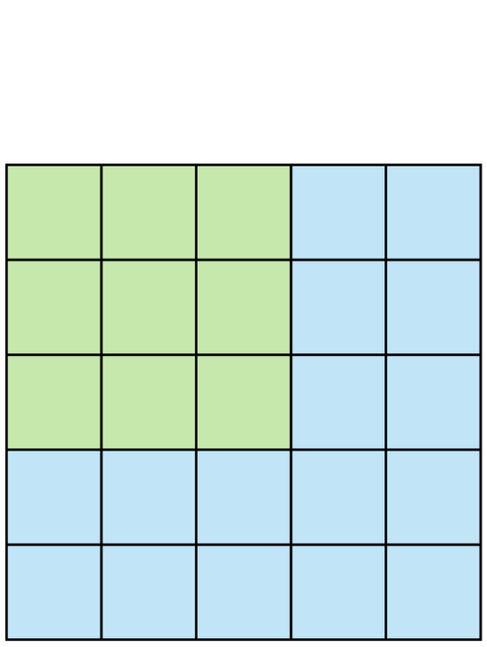
increasing stride from 1 to 2

Source: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

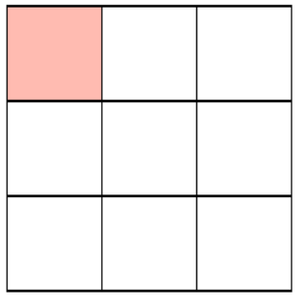
Padding

- The output of a convolution is valid as long as the summation uses available values
- In a convolution the valid output size is equal to: the input size - the size of the kernel + 1
- Unless we make boundary assumptions, a convolution will lead to a progressive shrinking of the input
- **Padding** is the assumption that outside the given domain the input takes some fixed values (e.g., zero)

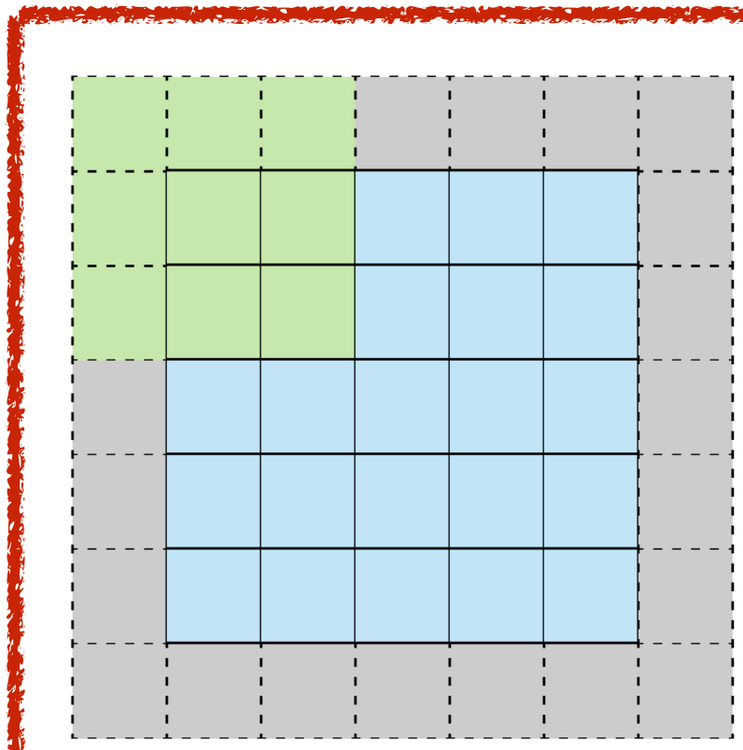
Padding Example



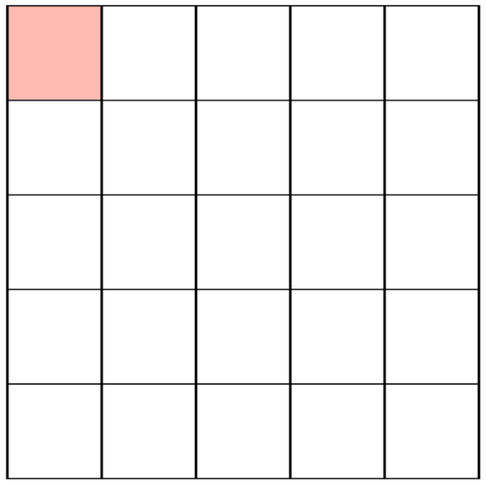
Stride 1



Feature Map



Stride 1 with Padding



Feature Map

adding zero padding of one pixel

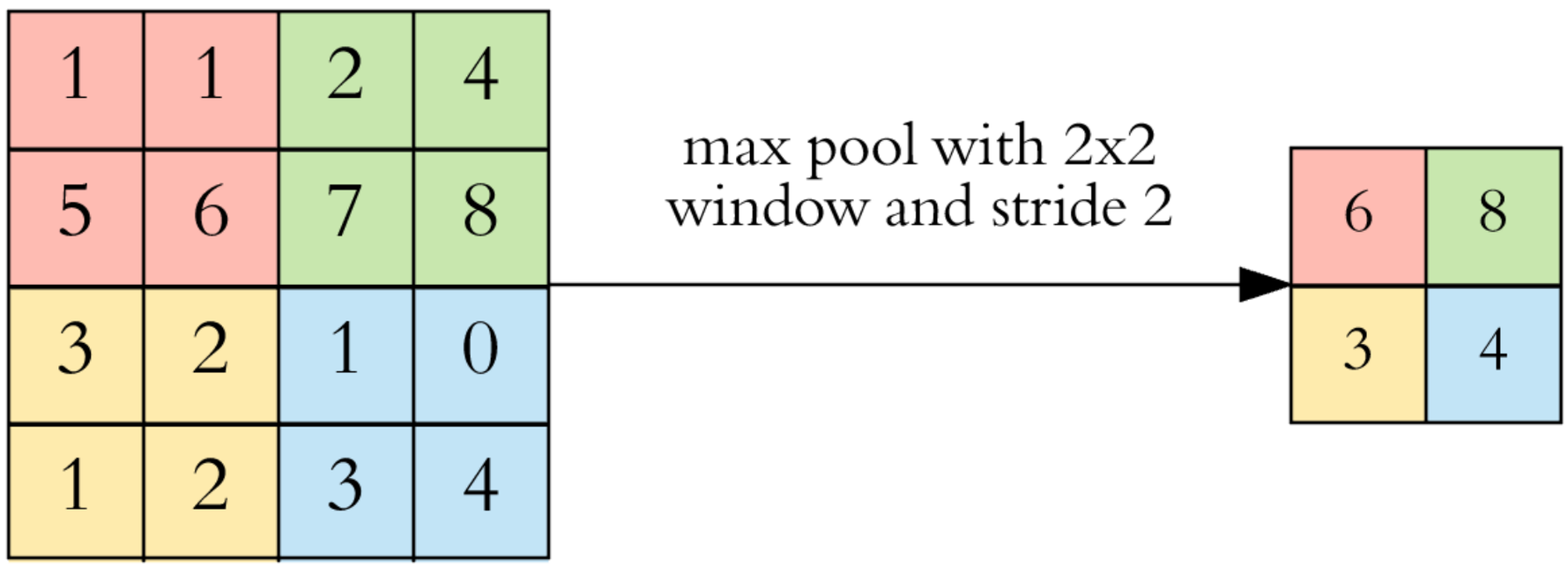
Source: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Pooling Layers

- The most common way to reduce the spatial dimension in classification tasks is max-pooling

- No learnable parameters

- Defined by window size, stride and padding



Data Types

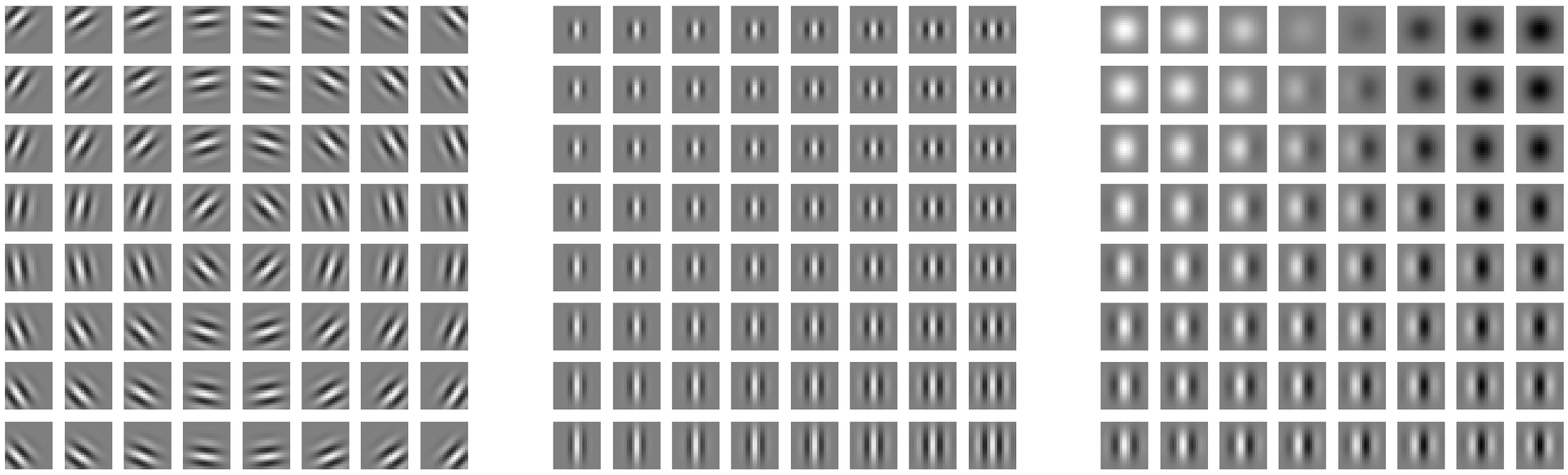
- Input data can be in different formats
- 1D: Audio waveforms (single channel) and skeleton animation data/motion (multi-channel)
- 2D: Audio data preprocessed via Fourier (single channel), color image data (multi-channel)
- 3D: Volumetric data such as CT scans (single channel), color video data (multi-channel)

Random or Unsupervised Features

- Kernels can be initialized
  - with random weights

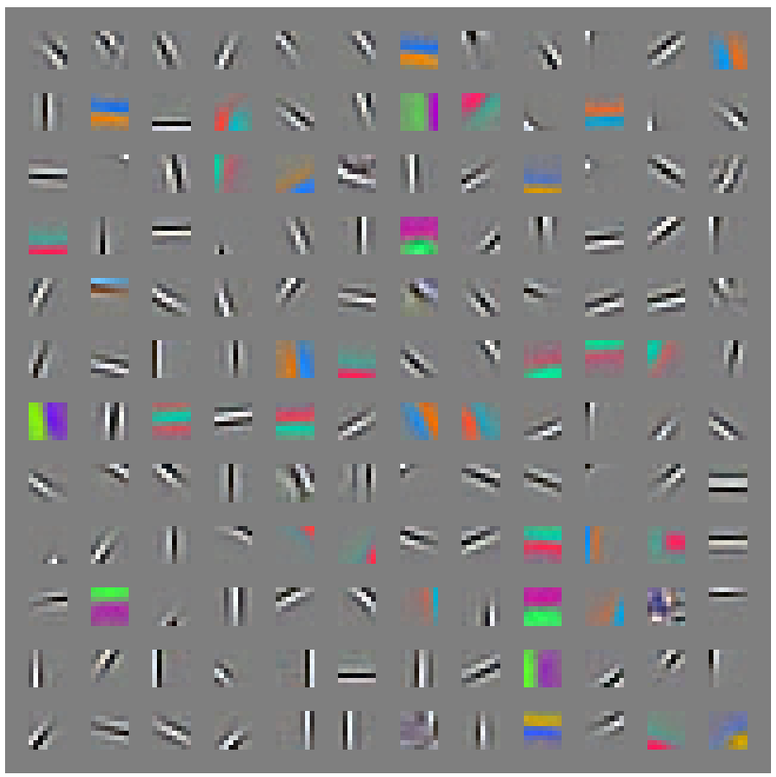
Random or Unsupervised Features

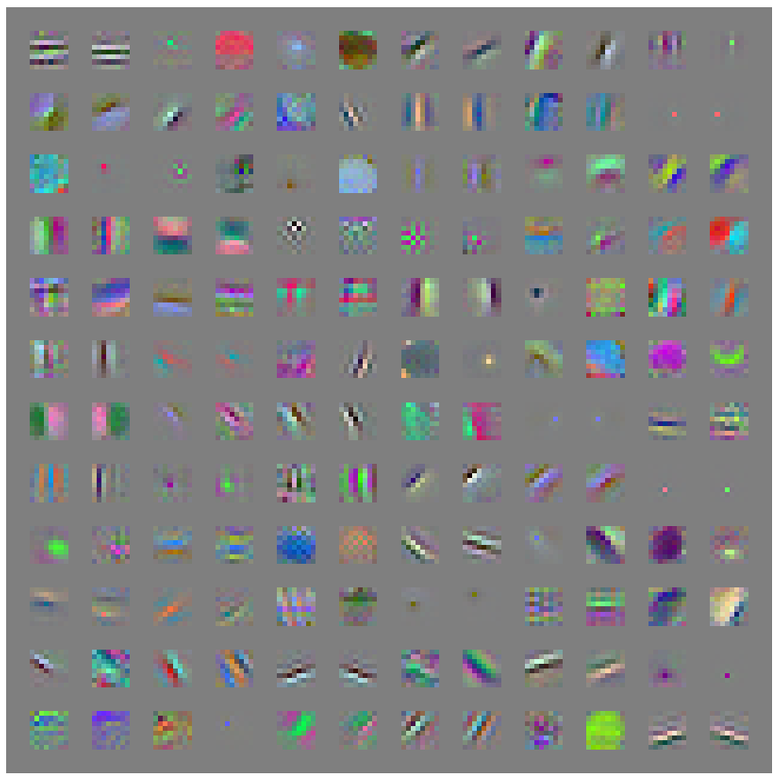
- Kernels can be initialized
  - with hand-designed features



Random or Unsupervised Features

- Kernels can be initialized
  - with unsupervised learning algorithms (e.g., apply k-means clustering to patches, then use centroids as kernels)





Thank you for your attention!

Questions?