A segmentation task in CMR

SCMR Series of Hands-on Workshops on Machine Learning

Saeed Karimi Bidhendi, PhD

Center for Pervasive Communications and Computing

University of California, Irvine

skarimib@uci.edu

Contents

- Recap
  - What is supervised learning?
  - Deep Neural Networks
  - Convolutional Neural Networks
  - DNN vs CNN: which one is better?
  - Training Neural Networks
- Introduction to segmentation in medical imaging
  - What is segmentation?
  - Which model should I use?
  - Which loss function is appropriate?
- Hands-on: Building a segmentation model
  - Data preprocessing
  - Model Architecture
  - Regularization
  - Hyperparameter Tuning
  - Training and Validation

Contents

- **Recap**
  - **What is supervised learning?**
  - Deep Neural Networks
  - Convolutional Neural Networks
  - DNN vs CNN: which one is better?
  - Training Neural Networks
- Introduction to segmentation in medical imaging
  - What is segmentation?
  - Which model should I use?
  - Which loss function is appropriate?
- Hands-on: Building a segmentation model
  - Data preprocessing
  - Model Architecture
  - Regularization
  - Hyperparameter Tuning
  - Training and Validation

What is supervised learning?

- Many applications demand inquiries about the data:
  - Given an image of a person, which patch contains the face? [Application: Camera Auto Focus]
  - Given an image of people, who are these people? [Application: Facebook Auto-tag System]
  - Given an email, is it malicious or not? [Application: Spam Filtering]
  - Given a voice command, which task is the user asking? [Application: Siri, Alexa, etc.]

- The goal of supervised learning is to build systems that would automate these tasks.

- How can a system learn to accomplish this?
  - Our only clue $\Longrightarrow$ Past observations of correct answers to exemplary inquiries:
  - The system should learn from these exemplary (data, label) pairs.

- The goal of supervised learning is to find a function $f(.)$ from data to labels, i.e. **function estimation**:

$$f(\boldsymbol{x}) = y$$

$\text{Email}_1 \rightarrow \text{Spam}$
$\text{Email}_2 \rightarrow \text{Not Spam}$
$\text{Email}_3 \rightarrow \text{Not Spam}$
$\text{Email}_4 \rightarrow \text{Not Spam}$
$\text{Email}_5 \rightarrow \text{Spam}$
$\vdots$

$\textbf{Data}(x_i) \xrightarrow{?} \textbf{Label}(y_i)$

Labeled Dataset

Training Dataset

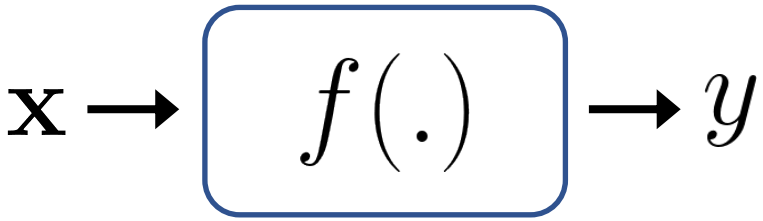
What is supervised learning?

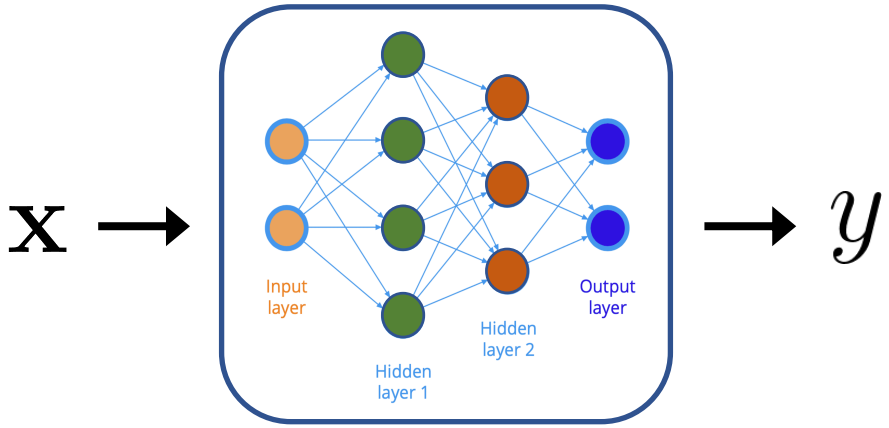
- Non-parametric Learning
  - The function $f(.)$ has no specific predefined parametric form.
  - Example: The k-nearest neighbors (KNN) algorithm
  - This is beyond the scope of this talk!

- Parametric Learning
  - The function $f(.)$ has a specific predefined parametric form.
  - Example: In linear regression, the function $f(.)$ has the following parametric form:

$$f(\boldsymbol{x}) = f(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{x}$$

  - The goal in parametric learning is to learn the optimal values of model's parameter.
  - In deep learning, the parametric function $f(.)$ is a neural networks.

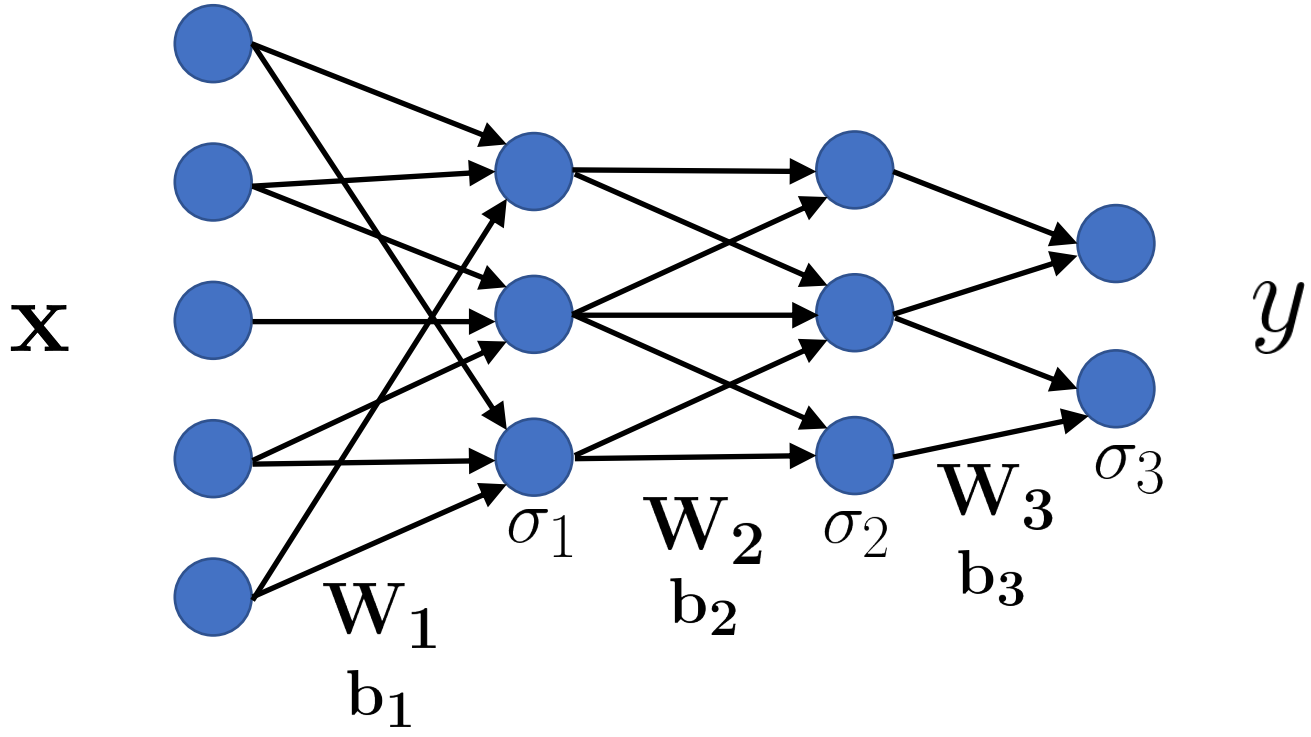




Contents

- **Recap**
  - What is supervised learning?
  - **Deep Neural Networks**
  - Convolutional Neural Networks
  - DNN vs CNN: which one is better?
  - Training Neural Networks
- Introduction to segmentation in medical imaging
  - What is segmentation?
  - Which model should I use?
  - Which loss function is appropriate?
- Hands-on: Building a segmentation model
  - Data preprocessing
  - Model Architecture
  - Regularization
  - Hyperparameter Tuning
  - Training and Validation

Deep Neural Networks

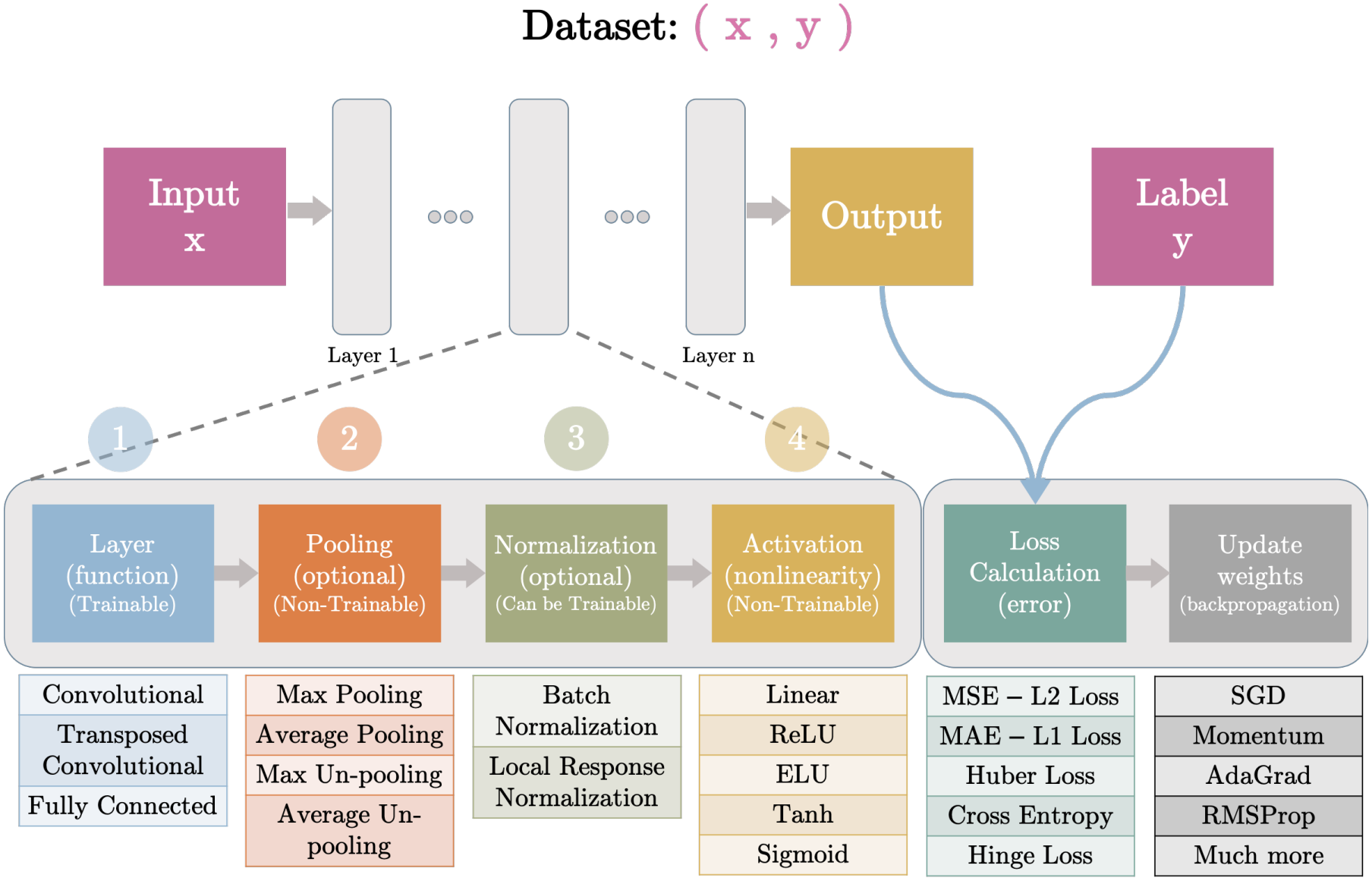


$$f(\boldsymbol{x}; \boldsymbol{W_1}, \boldsymbol{b_1}, \boldsymbol{W_2}, \boldsymbol{b_2}, \boldsymbol{W_3}, \boldsymbol{b_3}) = \sigma_3(\boldsymbol{W_3} \times \sigma_2(\boldsymbol{W_2} \times \sigma_1(\boldsymbol{W_1} \times \boldsymbol{x} + \boldsymbol{b_1}) + \boldsymbol{b_2}) + \boldsymbol{b_3})$$

Contents

- **Recap**
  - What is supervised learning?
  - Deep Neural Networks
  - **Convolutional Neural Networks**
  - DNN vs CNN: which one is better?
  - Training Neural Networks
- Introduction to segmentation in medical imaging
  - What is segmentation?
  - Which model should I use?
  - Which loss function is appropriate?
- Hands-on: Building a segmentation model
  - Data preprocessing
  - Model Architecture
  - Regularization
  - Hyperparameter Tuning
  - Training and Validation

Convolutional Neural Network



Source: Aqeel Anwar (aqeelanwarmalik)

Contents

- **Recap**
  - What is supervised learning?
  - Deep Neural Networks
  - Convolutional Neural Networks
  - **DNN vs CNN: which one is better?**
  - Training Neural Networks
- Introduction to segmentation in medical imaging
  - What is segmentation?
  - Which model should I use?
  - Which loss function is appropriate?
- Hands-on: Building a segmentation model
  - Data preprocessing
  - Model Architecture
  - Regularization
  - Hyperparameter Tuning
  - Training and Validation

DNN vs CNN: which one is better?

- No Free Lunch Theorem: There is no universally best model.
- Deep neural networks work well with vectorized data where features are permutation-invariant.
- Convolutional neural networks work well with image data that contain spatial information.
- The choice of the neural network and its architecture is application specific.

Contents

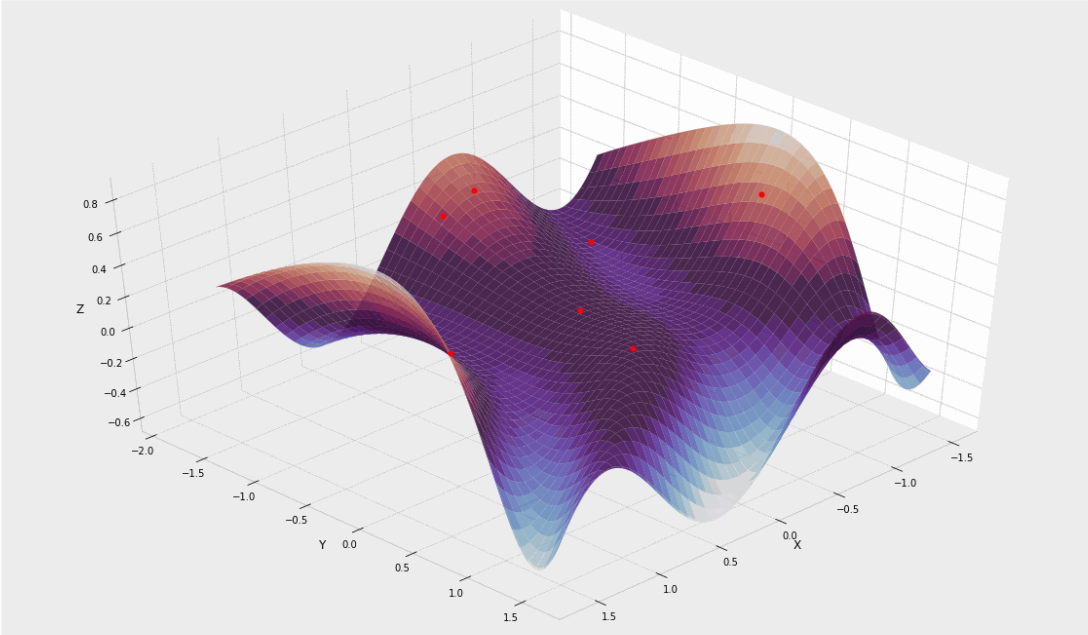
- **Recap**
  - What is supervised learning?
  - Deep Neural Networks
  - Convolutional Neural Networks
  - DNN vs CNN: which one is better?
  - **Training Neural Networks**
- Introduction to segmentation in medical imaging
  - What is segmentation?
  - Which model should I use?
  - Which loss function is appropriate?
- Hands-on: Building a segmentation model
  - Data preprocessing
  - Model Architecture
  - Regularization
  - Hyperparameter Tuning
  - Training and Validation

Training Neural Networks

- Gradient-based algorithms are used to learn model's parameters (weights and biases)
  - Stochastic Gradient Descent
  - Stochastic Gradient Descent with Momentum
  - Stochastic Gradient Descent with Nesterov Momentum
  - AdaGrad Algorithm
  - RMSProp Algorithm
  - RMSProp Algorithm with Nesterov Momentum
  - Adam Algorithm

$$\theta_{new} = \theta_{old} - \eta \nabla_{\theta} \text{Loss}$$

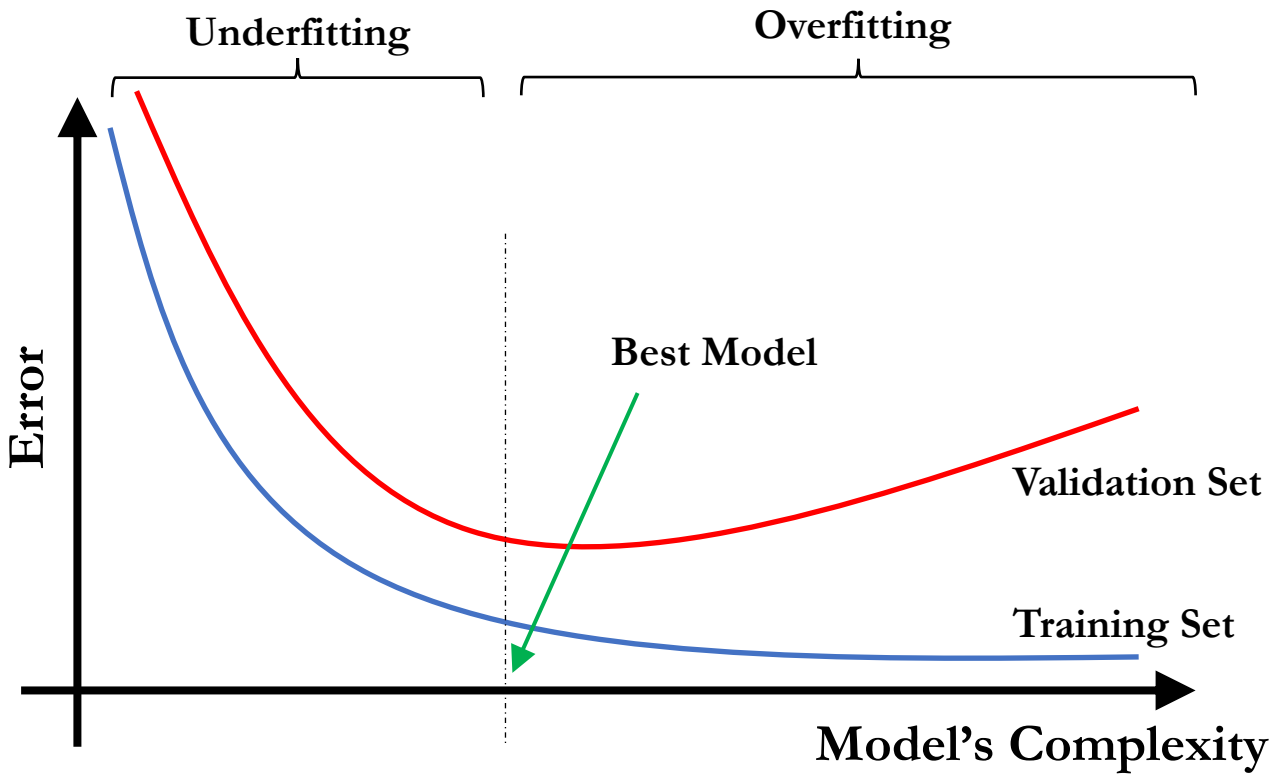
- Backpropagation Algorithm is used to calculate the gradients.



Source: https://www.kaggle.com/code/trolukovich/animating-gradien-descent/notebook

Training Neural Networks

- Generalizability:
  - Training Data → Training Data / Validation Data
  - Test Data
- Regularization:
  - Dropout
  - l^2 – Normalization
  - Early Stopping
  - Data Augmentation
  - ……



- A rule of thumb for choosing your model: Choose a high-complexity model and regularize it well!

Contents

- Recap
  - What is supervised learning?
  - Deep Neural Networks
  - Convolutional Neural Networks
  - DNN vs CNN: which one is better?
  - Training Neural Networks
- **Introduction to segmentation in medical imaging**
  - **What is segmentation?**
  - Which model should I use?
  - Which loss function is appropriate?
- Hands-on: Building a segmentation model
  - Data preprocessing
  - Model Architecture
  - Regularization
  - Hyperparameter Tuning
  - Training and Validation

What is segmentation?

- Image segmentation is referred to the task of clustering parts of an image that are from the same object class.

- Image segmentation can also be viewed as **pixel-level classification**.

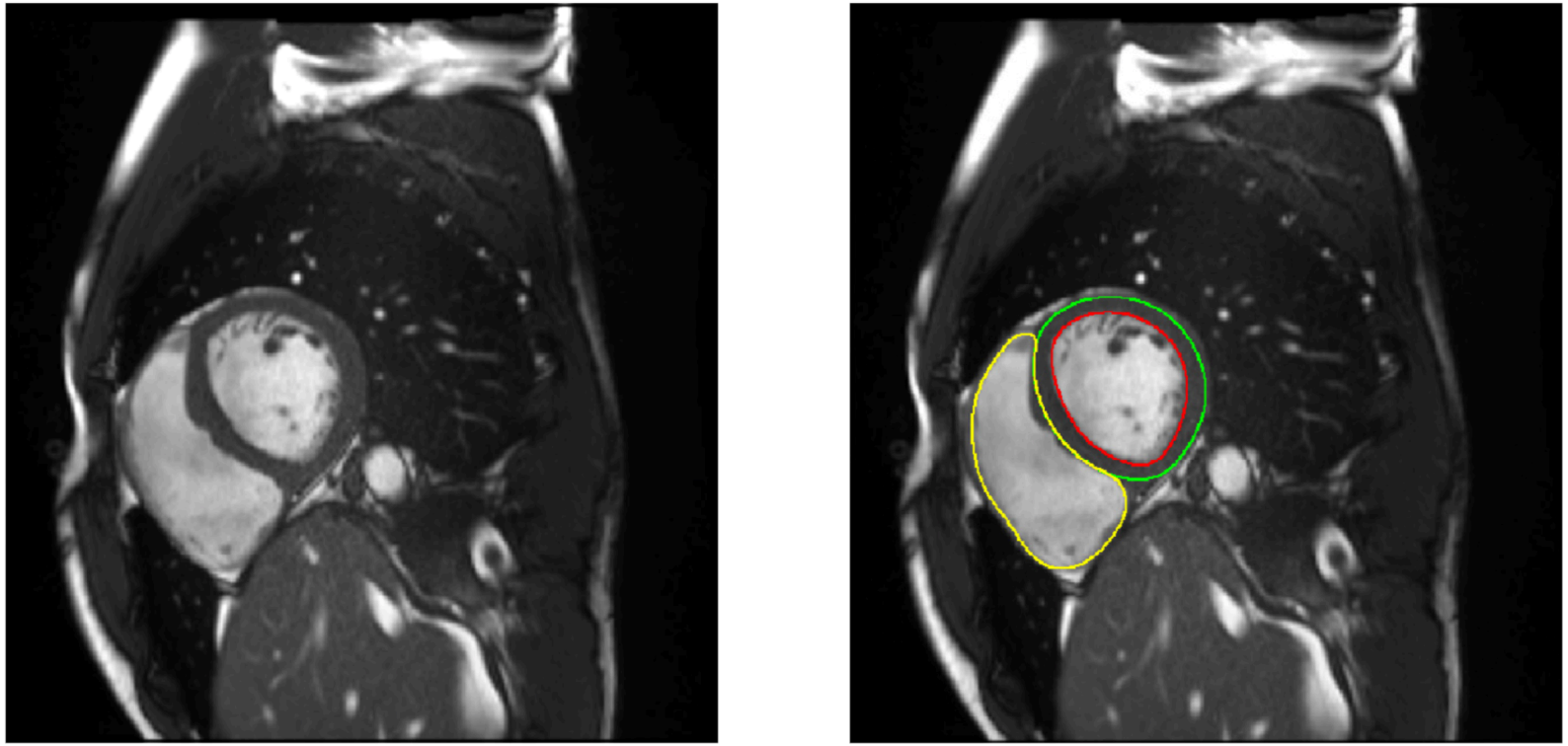
- Both data and label are images of same dimension.

Each pixel should be classified as either dog, cat, or background.

https://github.com/parachutel/deeplabv3plus_on_Mapillary_Vistas

What is segmentation?

- Applications in CMR:



A slice from the MRI scans before and after segmentation. Yellow contour: right endocardium, red: left endocardium, green: left epicardium.

- See bwconncomp and bwboundaries for extracting contour from a binary mask.

Source: Budai A, Suhai FI, Csorba K, Toth A, Szabo L, Vago H, Merkely B. Fully automatic segmentation of right and left ventricle on short-axis cardiac MRI images. Computerized Medical Imaging and Graphics. 2020 Oct 1;85:101786.

What is segmentation?

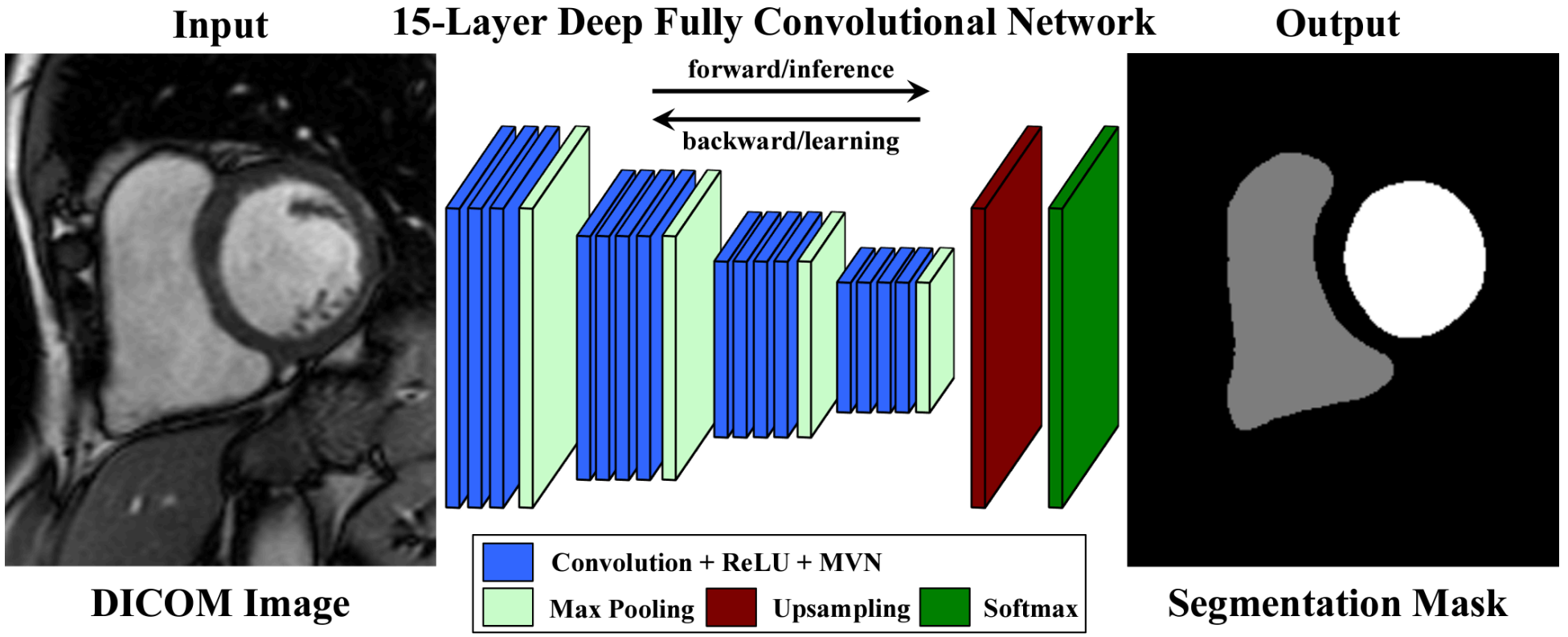
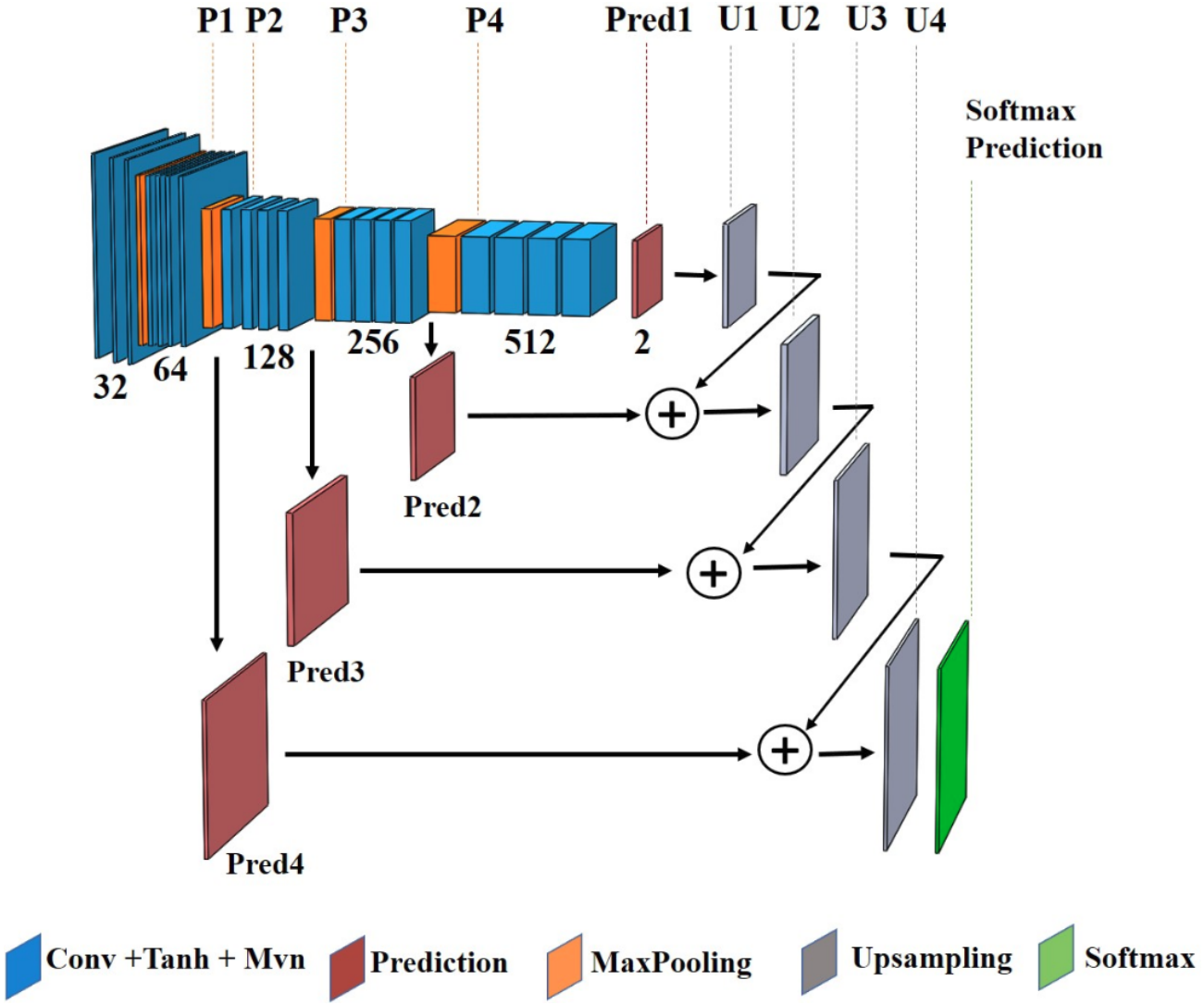


Figure 1. A schematic of our proposed fully convolutional neural network architecture. Acronyms: ReLU – Rectified Linear Unit; MVN – Mean-Variance Normalization.

Tran PV. A fully convolutional neural network for cardiac segmentation in short-axis MRI. arXiv preprint arXiv:1604.00494. 2016 Apr 2.

What is segmentation?



Source: Karimi-Bidhendi S, Arafati A, Cheng AL, Wu Y, Kheradvar A, Jafarkhani H. Fully-automated deep-learning segmentation of pediatric cardiovascular magnetic resonance of patients with complex congenital heart diseases. Journal of cardiovascular magnetic resonance. 2020 Dec;22(1):1-24.

Contents

- Recap
  - What is supervised learning?
  - Deep Neural Networks
  - Convolutional Neural Networks
  - DNN vs CNN: which one is better?
  - Training Neural Networks
- **Introduction to segmentation in medical imaging**
  - What is segmentation?
  - **Which model should I use?**
  - Which loss function is appropriate?
- Hands-on: Building a segmentation model
  - Data preprocessing
  - Model Architecture
  - Regularization
  - Hyperparameter Tuning
  - Training and Validation

Which model should I use?

- Since CMR data are images, it is better to start with convolutional neural networks.
- Deeper networks that are well regularized usually work better.
- Dropout, Early Stopping, and Data Augmentation are good default regularization methods.
- ReLU is usually a good default activation function.
- The choice of loss function should represent our performance metric (e.g. dice, cross-entropy, …)
- Skip-connections can help with network's training.
- Adam optimizer is a good default choice for network's training.

Contents

- Recap
  - What is supervised learning?
  - Deep Neural Networks
  - Convolutional Neural Networks
  - DNN vs CNN: which one is better?
  - Training Neural Networks
- **Introduction to segmentation in medical imaging**
  - What is segmentation?
  - Which model should I use?
  - **Which loss function is appropriate?**
- Hands-on: Building a segmentation model
  - Data preprocessing
  - Model Architecture
  - Regularization
  - Hyperparameter Tuning
  - Training and Validation

Which loss function is appropriate?

- Cross-Entropy is a good default loss function since it is closely related to the principle of maximum likelihood estimation.
- Mean square error is usually not a good default choice for classification tasks.
- Task related cost functions can be more effective for the desired performance metric.

Contents

- Recap
  - What is supervised learning?
  - Deep Neural Networks
  - Convolutional Neural Networks
  - DNN vs CNN: which one is better?
  - Training Neural Networks
- Introduction to segmentation in medical imaging
  - What is segmentation?
  - Which model should I use?
  - Which loss function is appropriate?
- **Hands-on: Building a segmentation model**
  - **Data preprocessing**
  - Model Architecture
  - Regularization
  - Hyperparameter Tuning
  - Training and Validation

Data preprocessing

- Raw data is in the form of DICOM (Digital Imaging and Communications in Medicine).

- CMR images should be extracted from DICOM files.
  - Python commands to extract DICOM files:

```
>>> from pydicom import dcmread
>>> ds = dcmread('Sample_Dicom.dcm')
>>> image = ds.pixel_array
>>> print('Image shape = {0}'.format(image.shape))
Image shape = (384, 384)
>>>
```

  - MATLAB commands to extract DICOM files:

```
Command Window
>> image = dicomread('Sample_Dicom.dcm');
>> size(image)

ans =

   384   384

>>
```

Data preprocessing

- Meta data information should also be extracted from DICOM files:
  - Pixel Spacing ⟹ Tag = (0028, 0030)
  - Spacing Between Slices ⟹ Tag = (0018, 0088)
  - Slice Thickness ⟹ Tag = (0018, 0050)
  - ……

  - Meta data information extraction in Python:

```
>>> ds.SpacingBetweenSlices
"8"
>>> element = ds[0x0018, 0x0088]
>>> element.value
"8"
```

  - Meta data information extraction in MATLAB:

```
Command Window
>> ds = dicominfo('Sample_Dicom.dcm');
>> ds.SpacingBetweenSlices

ans =

     8

fx >>
```

Data preprocessing

- CMR images should be prepared for network's training:
  - The size of images should be adjusted according to the input layer of the neural network.
  - All training images are usually assembled into a fourth-order tensor of the following shape:
    $(\text{number of images},\ \text{image width},\ \text{image height},\ \text{number of channels})$, e.g. $(1000, 128, 128, 1)$
  - The training set should be split into training/validation sets. Common ratios: $70\% - 30\%$ and $80\% - 20\%$.
  - Data normalization can have significant effect on network's performance based on the application, network, etc.
  - Data augmentation can significantly improve the network's performance:
    - Data augmentation can be done offline or on-the-fly.
    - Exemplary techniques: rotation, horizontal and vertical flipping, adding noise, cropping, changing contrast, etc.

```
from keras.preprocessing.image import ImageDataGenerator
train_datagen = ImageDataGenerator(rotation_range=180, horizontal_flip=True, vertical_flip=True,
                                   featurewise_std_normalization=True, validation_split=0.3)
# Compute quantities required for featurewise normalization
# (std, mean, and principal components if ZCA whitening is applied)
train_datagen.fit(training_images)
```

Contents

- Recap
  - What is supervised learning?
  - Deep Neural Networks
  - Convolutional Neural Networks
  - DNN vs CNN: which one is better?
  - Training Neural Networks
- Introduction to segmentation in medical imaging
  - What is segmentation?
  - Which model should I use?
  - Which loss function is appropriate?
- **Hands-on: Building a segmentation model**
  - Data preprocessing
  - **Model Architecture**
  - Regularization
  - Hyperparameter Tuning
  - Training and Validation

Model Architecture

- We manually define the Dice coefficient loss as our objective function:
  - We want to maximize the Dice coefficient defined as: $\mathrm{Dice}(\hat{Y}, Y) = \frac{2|\hat{Y} \cap Y|}{|\hat{Y}| + |Y|}$
  - Alternatively, we want to minimize the negative Dice coefficient: $\text{Loss function} = -\mathrm{Dice}(\hat{Y}, Y)$

```
from keras import backend as K
# Average dice coefficient per batch
def dice_coef(y_true, y_pred, smooth=0.0):
    """
    Average dice coefficient per batch
    :param y_true: True labels
    :param y_pred: Predicted labels
    :param smooth: a scalar (default to 0.0)
    :return: the Dice index
    """
    axes = (1, 2, 3)
    intersection = K.sum(y_true * y_pred, axis=axes)
    summation = K.sum(y_true, axis=axes) + K.sum(y_pred, axis=axes)

    return K.mean((2.0 * intersection + smooth) / (summation + smooth), axis=0)

# Objective loss function to minimize, which is 1 minus the dice index
def dice_coef_loss(y_true, y_pred):
    return 1.0 - dice_coef(y_true, y_pred, smooth=0.1)
```

Model Architecture

- The model consists of a series of convolution and pooling layers followed by the choice of optimizer, loss function, etc.

```
# A simple U-Net model for image segmentation
def unet(pretrained_weights = None,input_size = (128, 128, 1)):
    inputs = Input(input_size)
    conv1 = Conv2D(2, 3, activation = 'tanh', padding = 'same', kernel_initializer = 'glorot_uniform')(inputs)
    conv1 = Conv2D(2, 3, activation = 'tanh', padding = 'same', kernel_initializer = 'glorot_uniform')(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
    conv2 = Conv2D(4, 3, activation = 'tanh', padding = 'same', kernel_initializer = 'glorot_uniform')(pool1)
    conv2 = Conv2D(4, 3, activation = 'tanh', padding = 'same', kernel_initializer = 'glorot_uniform')(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
    conv3 = Conv2D(8, 3, activation = 'tanh', padding = 'same', kernel_initializer = 'glorot_uniform')(pool2)
    conv3 = Conv2D(8, 3, activation = 'tanh', padding = 'same', kernel_initializer = 'glorot_uniform')(conv3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)
```

⋮

```
conv9 = Conv2D(2, 3, activation = 'tanh', padding = 'same', kernel_initializer = 'glorot_uniform')(conv9)
output = Conv2D(1, 1, activation = 'sigmoid')(conv9)

model = Model(inputs, output)

model.compile(optimizer = Adam(learning_rate = 0.001), loss = dice_coef_loss, metrics = [dice_coef])

if(pretrained_weights):
  model.load_weights(pretrained_weights)

return model
```

Contents

- Recap
  - What is supervised learning?
  - Deep Neural Networks
  - Convolutional Neural Networks
  - DNN vs CNN: which one is better?
  - Training Neural Networks
- Introduction to segmentation in medical imaging
  - What is segmentation?
  - Which model should I use?
  - Which loss function is appropriate?
- **Hands-on: Building a segmentation model**
  - Data preprocessing
  - Model Architecture
  - **Regularization**
  - Hyperparameter Tuning
  - Training and Validation

Regularization

- Dropout can be applied as a safe default regularization method.
  - In each iteration, a random set of neurons are removed (equivalently, these neurons are multiplied by zero).
  - Dropout trains an ensemble consisting of sub-networks that are sampled from the base network.
  - There are exponentially many sub-networks.
  - Training these many networks is feasible since models share parameters.

```
conv4 = Conv2D(16, 3, activation = 'tanh', padding = 'same', kernel_initializer = 'glorot_uniform')(conv4)
drop4 = Dropout(0.5)(conv4)
pool4 = MaxPooling2D(pool_size=(2, 2))(drop4)

conv5 = Conv2D(32, 3, activation = 'tanh', padding = 'same', kernel_initializer = 'glorot_uniform')(pool4)
conv5 = Conv2D(32, 3, activation = 'tanh', padding = 'same', kernel_initializer = 'glorot_uniform')(conv5)
drop5 = Dropout(0.5)(conv5)
```

Regularization

- Other regularization techniques are also available for layer's weights, biases, activations, etc.

  - $\boldsymbol{l^2 - \text{regularization}}$ $\boldsymbol{(\text{weight decay})}$: It derives the weights closer to the origin by penalizing the square norm of parameters, i.e., the regularization term $\lambda\|\boldsymbol{W}\|_2^2$ is added to the original objective function:

$$\hat{J}(\boldsymbol{\Theta}; \boldsymbol{X}, \boldsymbol{y}) = J(\boldsymbol{\Theta}; \boldsymbol{X}, \boldsymbol{y}) + \lambda\|\boldsymbol{W}\|_2^2$$

  - $\boldsymbol{l^1 - \text{regularization}}$: It encourages the solution to be sparse, i.e., some parameters having an optimal value of zero, by adding the penalty term $\lambda\|\boldsymbol{W}\|_1 = \lambda\sum_i |W_i|$

Regularization

- Early Stopping:
  - The algorithm terminates, when no parameter is improved over the best recorded validation error.
  - Early stopping can be viewed as an efficient hyperparameter selection technique:
    - The number of training iterations is considered a hyperparameter.
    - We are controlling model's effective capacity by specifying the number of training steps to fit the training set.
    - Instead of guess and check, this hyperparameter is set in a single training run.
    - It comes with the additional cost of tracing validation error in each iteration.
  - Early stopping restricts the optimization to a relatively small volume of parameter space in the neighborhood of the initial parameter values since the number of training updates is restricted.
  - Early stopping requires little to no modification to the training pipeline; hence, it is the most common regularization.
  - Early stopping can be used alone or along with other regularization techniques.

Contents

- Recap
  - What is supervised learning?
  - Deep Neural Networks
  - Convolutional Neural Networks
  - DNN vs CNN: which one is better?
  - Training Neural Networks
- Introduction to segmentation in medical imaging
  - What is segmentation?
  - Which model should I use?
  - Which loss function is appropriate?
- **Hands-on: Building a segmentation model**
  - Data preprocessing
  - Model Architecture
  - Regularization
  - **Hyperparameter Tuning**
  - Training and Validation

Hyperparameter Tuning

- Hyperparameters control the behavior of the algorithm such as its runtime, memory cost, quality of the trained model, and etc. Examples include
  - Learning rate
  - Dropout rate
  - Number of hidden layers
  - Number of nodes in each hidden layer
  - The choice of activation functions
  - The convolution kernel width and stride
  - ……

- There are several principles for hyperparameter selection:
  - Manual hyperparameter tuning
  - Grid Search
  - Random search
  - ……

Hyperparameter Tuning

- One intuition for hyperparameter selection is **how it affects the model's effective capacity**:
  - Increasing the number of hidden layers $\Longrightarrow$ Model's capacity is increased
  - Increasing the number of hidden units $\Longrightarrow$ Model's capacity is increased
  - Decreasing the Dropout rate $\Longrightarrow$ Model's capacity is increased
  - Increasing the convolution kernel width $\Longrightarrow$ Model's capacity is increased
  - Optimally-tuned learning rate $\Longrightarrow$ Model's capacity is increased
  - ……

- Grid search is most effective when values are selected on a logarithmic scale:
  - Learning rate taking values of $0.1, 0.01, 10^{-3}, 10^{-4}, 10^{-5}$
  - Number of hidden units taken from the set $\{50, 100, 200, 500, 1000, 2000, 4000\}$

- Learning rate is the most important hyperparameter to tune.

Contents

- Recap
  - What is supervised learning?
  - Deep Neural Networks
  - Convolutional Neural Networks
  - DNN vs CNN: which one is better?
  - Training Neural Networks
- Introduction to segmentation in medical imaging
  - What is segmentation?
  - Which model should I use?
  - Which loss function is appropriate?
- **Hands-on: Building a segmentation model**
  - Data preprocessing
  - Model Architecture
  - Regularization
  - Hyperparameter Tuning
  - **Training and Validation**

Training and Validation

- Recap:
  - Preprocessing the data
  - Choosing the model's architecture
  - Incorporate the regularization method(s)
  - Tune the hyperparameters
  - Selecting the optimization method
  - Fit the model to the data and monitor the validation error
  - Use the trained model for performance evaluation on test data

- It is a good practice to create checkpoints and callbacks for saving the best model

Let's go see a hands-on example!

Training and Validation

- Source of data and study:
  - https://github.com/saeedkarimi/Cardiac_MRI_Segmentation
  - Fully-automated deep-learning segmentation of pediatric cardiovascular magnetic resonance of patients with complex congenital heart diseases

Thank you for your attention!

Olivier Jaubert

Javier Montalt-Tordera

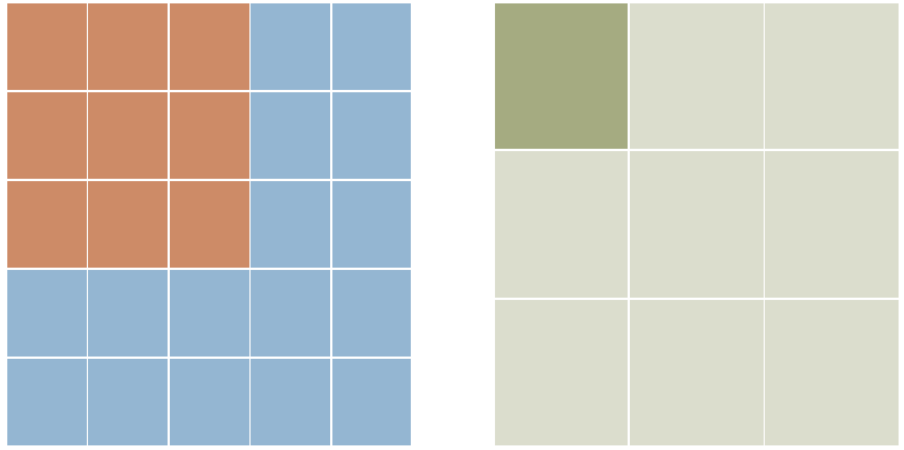
Vivek Muthurangu

Jennifer Steeden

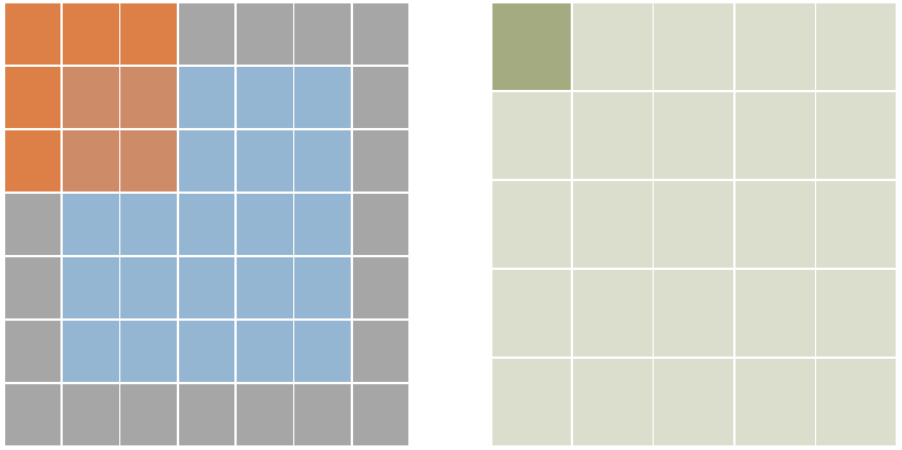
Society for Cardiovascular Magnetic Resonance

Convolutional Neural Network

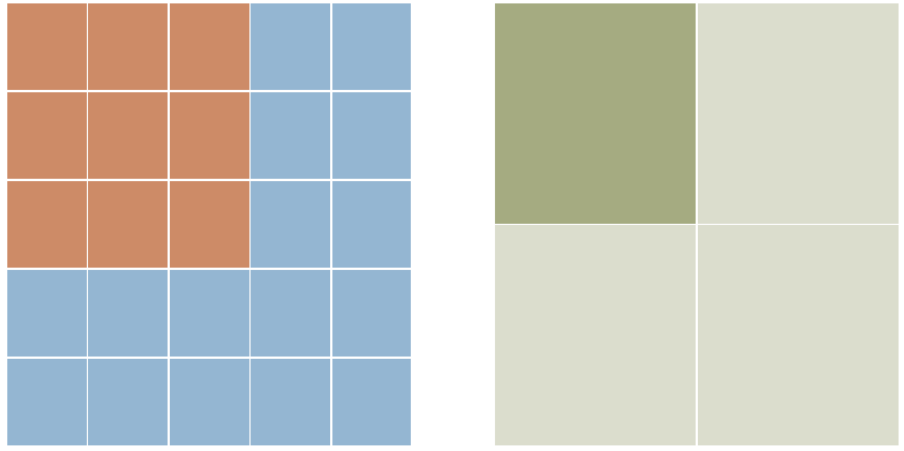
Type: conv - Stride: 1 Padding: 0



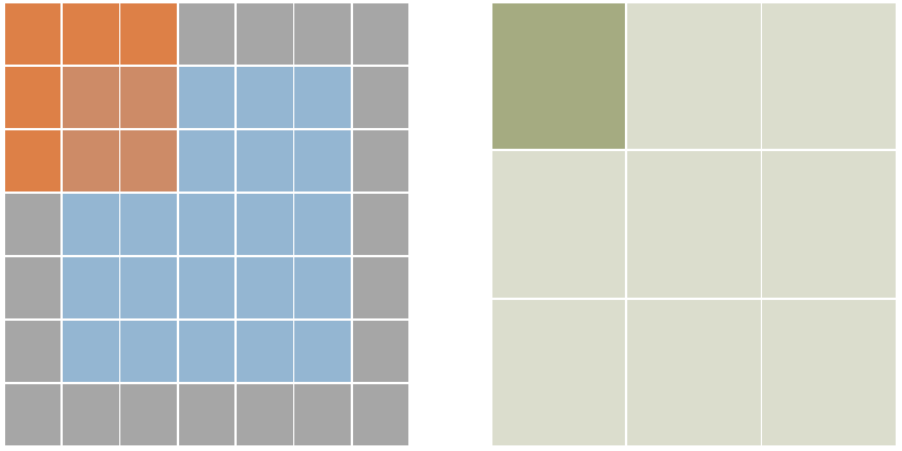
Type: conv - Stride: 1 Padding: 1



Type: conv - Stride: 2 Padding: 0

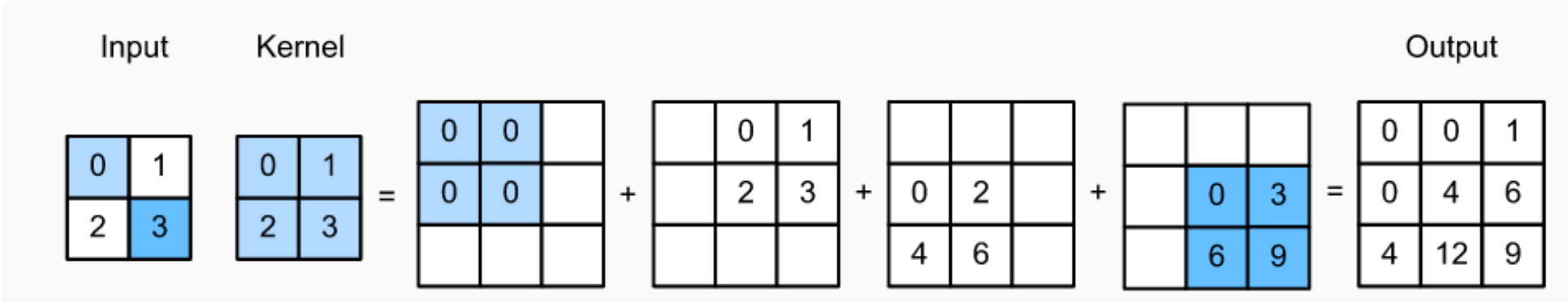


Type: conv - Stride: 2 Padding: 1



Source: https://towardsdatascience.com/a-visualization-of-the-basic-elements-of-a-convolutional-neural-network-75fea30cd78d

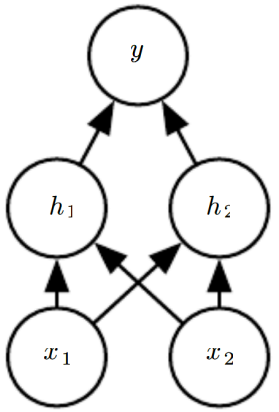
Transposed Convolution



Source of the image: https://towardsdatascience.com/transposed-convolution-demystified-84ca81b4baba

Regularization

- Dropout can be applied as a safe default regularization method.
  - In each iteration, a random set of neurons are removed (equivalently, these neurons are multiplied by zero).
  - Dropout trains an ensemble consisting of sub-networks that are sampled from the base network.

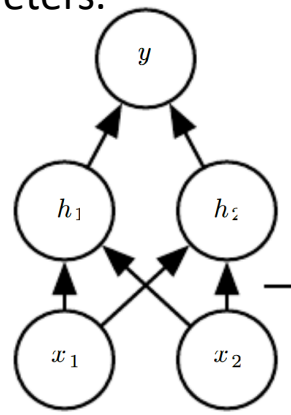


Base network

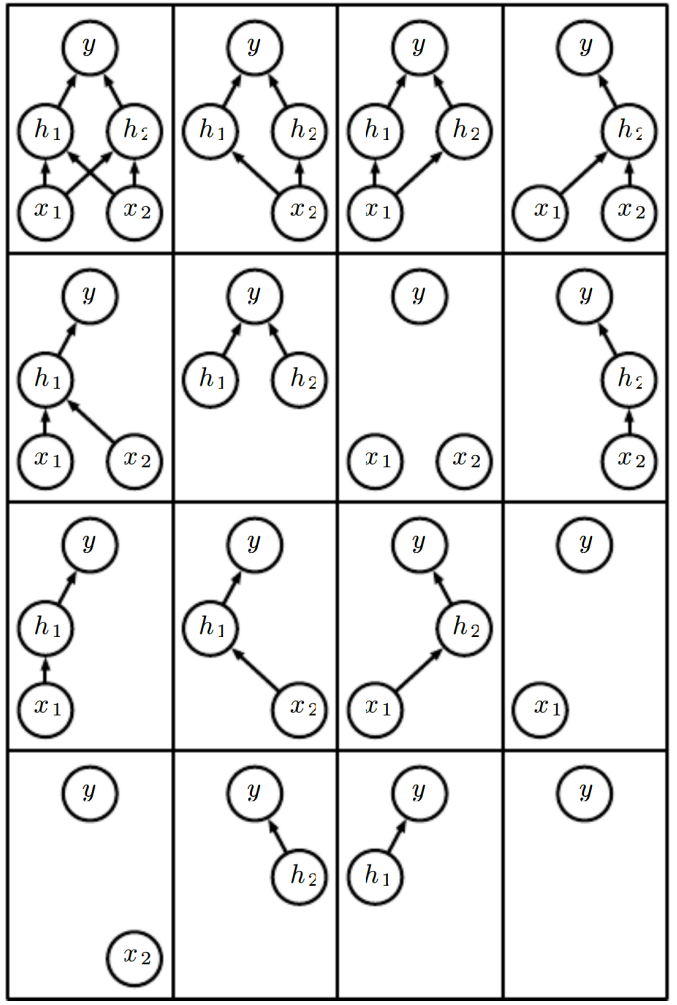
Source: Goodfellow I, Bengio Y, Courville A. Deep learning. MIT press; 2016 Nov 10.

Regularization

- Dropout can be applied as a safe default regularization method.
  - In each iteration, a random set of neurons are removed (equivalently, these neurons are multiplied by zero).
  - Dropout trains an ensemble consisting of sub-networks that are sampled from the base network.
  - There are exponentially many sub-networks.
  - Training these many networks is feasible since models share parameters.



Base network



Ensemble of subnetworks

```
conv4 = Conv2D(16, 3, activation = 'tanh', padding = 'same', kernel_initializer = 'glorot_uniform')(conv4)
drop4 = Dropout(0.5)(conv4)
pool4 = MaxPooling2D(pool_size=(2, 2))(drop4)

conv5 = Conv2D(32, 3, activation = 'tanh', padding = 'same', kernel_initializer = 'glorot_uniform')(pool4)
conv5 = Conv2D(32, 3, activation = 'tanh', padding = 'same', kernel_initializer = 'glorot_uniform')(conv5)
drop5 = Dropout(0.5)(conv5)
```

Source: Goodfellow I, Bengio Y, Courville A. Deep learning. MIT press; 2016 Nov 10.

ZCA Whitening

- It is an image preprocessing technique that leads to a transformation of data such that the covariance matrix is the identity matrix, yielding decorrelated features.

- Eigen decomposition of covariance matrix:

$$C = PDP^T$$

- ZCA-Whitening transformation matrix:

$$W_{ZCA} = PD^{-\frac{1}{2}}P^T = C^{-\frac{1}{2}}$$

$$\text{ZCA whitened data} = W_{ZCA} \times \text{original data}$$