Institut et hôpital neurologiques de Montréal
Montreal Neurological Institute and Hospital

Deep Learning with MRI

Neurotech MTL

McGill | Centre universitaire de santé McGill McGill University Health Centre

About Us

Thomas Funck: Having originally studied philosophy and cognitive science as an undergraduate at McGill, Thomas is now a Ph.D. candidate in neuroscience at the MNI. He uses multi-modal brain imaging, signal processing, and computational simulation to study the cellular architecture of the living brain.
Twitter : @tffunck

Andrew Doyle:
During his MSc with Tal Arbel, Andrew developed machine learning algorithms to detect lesions in multiple sclerosis. He is now continuing his work in the MCIN lab at the MNI, with a special interest in using AI to automate quality control for brain imaging.
Twitter : @crocodoyle

Robert Fratila: Robert is a Co-Founder and CTO of aifred health. He is very passionate about combining medicine and AI throughout his work and research. His work as a software developer at the Montreal Neurological Institute, integrating state-of-the-art machine learning models in healthcare, specifically brain imaging, has given him lots of experience in finding efficient solutions to complex problems.
LinkedIn : robertfratila

neuro

McGill | Centre universitaire de santé McGill | McGill University Health Centre

Installation

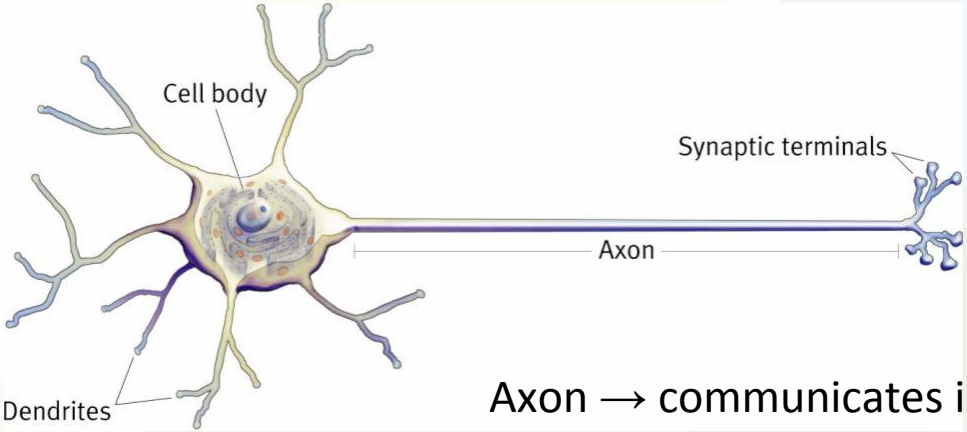
- Google Colab (super easy)
  - Create / Log-in to Google account
  - Go to https://colab.research.google.com
  - Download and load: https://tinyurl.com/yd8dd5x3
- Docker (very easy):
  - Install docker on your OS
    - https://docs.docker.com/install/#cloud
  - docker pull tffunck/neurotech:latest
  - docker run -it --rm tffunck/neurotech:latest
- DIY (pretty easy):
  - wget https://bootstrap.pypa.io/get-pip.py
    - Or go to the link and download manually
  - python3 get-pip.py
  - pip3 install pandas numpy scipy h5py tensorflow keras
  - git clone https://github.com/tfunck/minc_keras
- Data
  - Unzip : minc_keras/data/ouput.gz.bz2

Outline

1. Brain imaging and neuroanatomy
2. Machine Learning
3. Deep Learning
4. Keras Example

Super simple neuroanatomy

Cell body and dendrites → integrate information

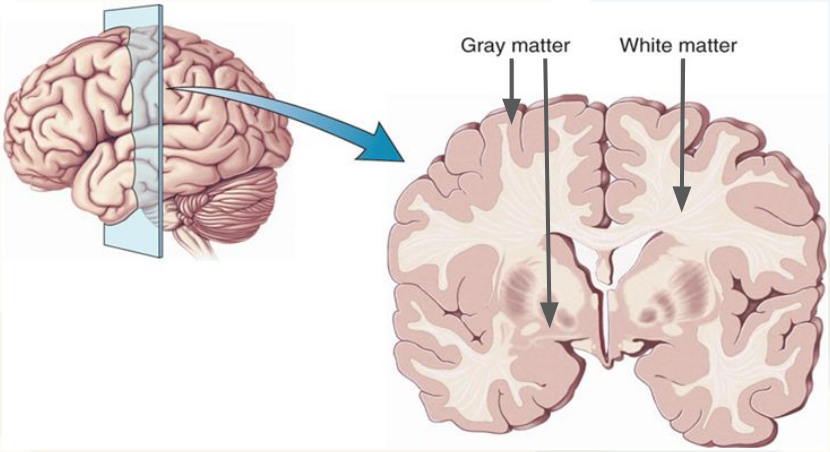


Axon → communicates information downstream

neuro

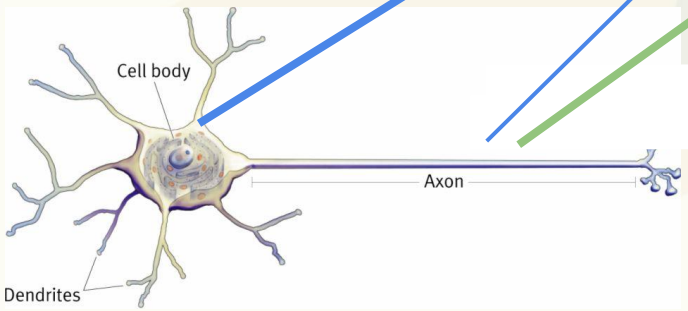
McGill | Centre universitaire de santé McGill | McGill University Health Centre

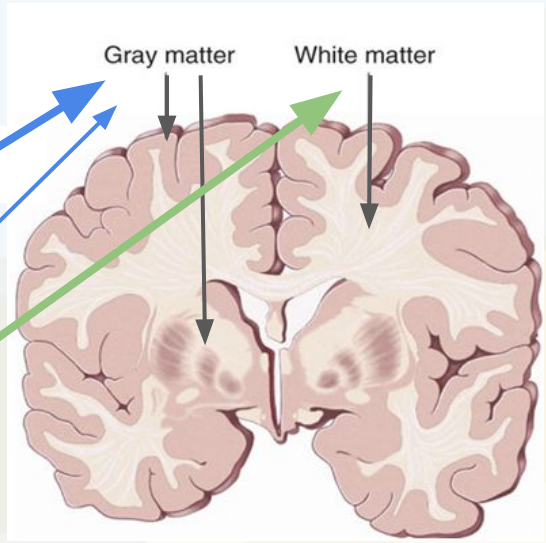
Super simple neuroanatomy



Super simple neuroanatomy

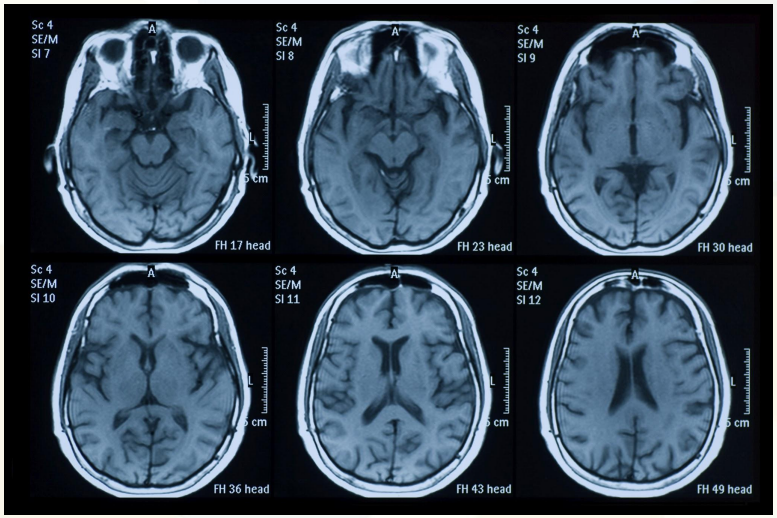
Grey Matter (GM) : cell bodies + short range axons
White Matter (WM) : long range axons
Cerebrospinal Fluid (CSF) : liquid that your brain floats in





Magnetic Resonance Imaging (MRI)

- Uses powerful magnets to create images of biological tissue (e.g., brains)

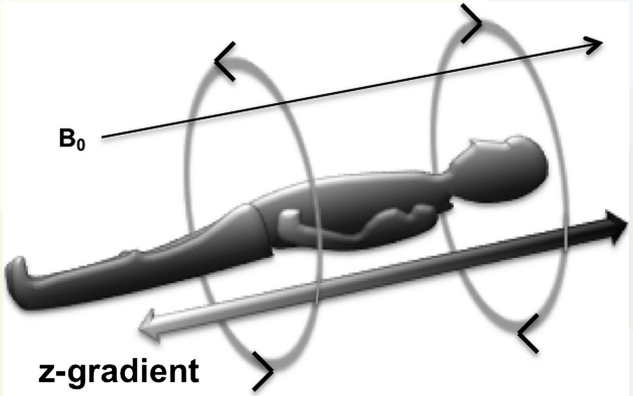


neuro

McGill | Centre universitaire de santé McGill | McGill University Health Centre

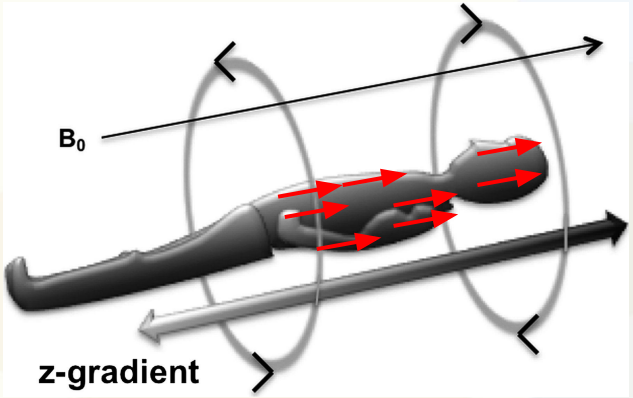
Magnetic Resonance Imaging (MRI)

1. MRI scanners first create a magnetic field along the axis of the scanner



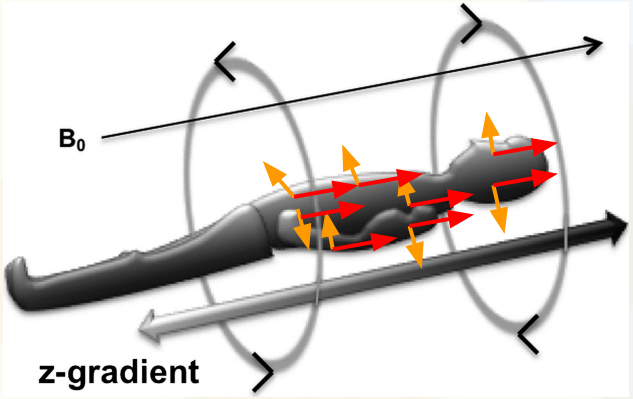
Magnetic Resonance Imaging (MRI)

1. MRI machines first create a magnetic field along the axis of the scanner
2. Aligns hydrogen (H) atoms in the body to the magnetic field



Magnetic Resonance Imaging (MRI)

1. MRI machines first create a magnetic field along the axis of the scanner
2. Aligns hydrogen (H) atoms in the body to the magnetic field
3. MRI emits magnetic pulse that knocks H atoms out of alignment



neuro

McGill | Centre universitaire de santé McGill | McGill University Health Centre

Magnetic Resonance Imaging (MRI)

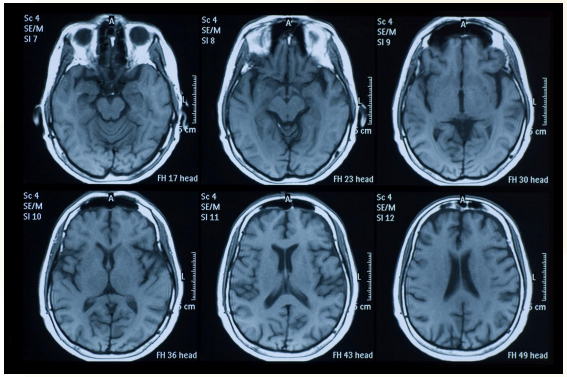
1. MRI machines first create a magnetic field along the axis of the scanner
2. Aligns hydrogen (H) atoms in the body to the magnetic field
3. MRI emits magnetic pulse that knocks H atoms out of alignment
4. Time it takes for H atoms to regain alignment depends on biological tissue

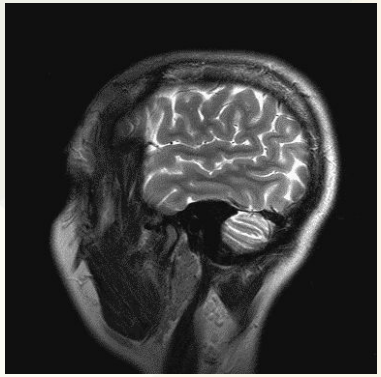
neuro

McGill | Centre universitaire de santé McGill | McGill University Health Centre

Magnetic Resonance Imaging (MRI)

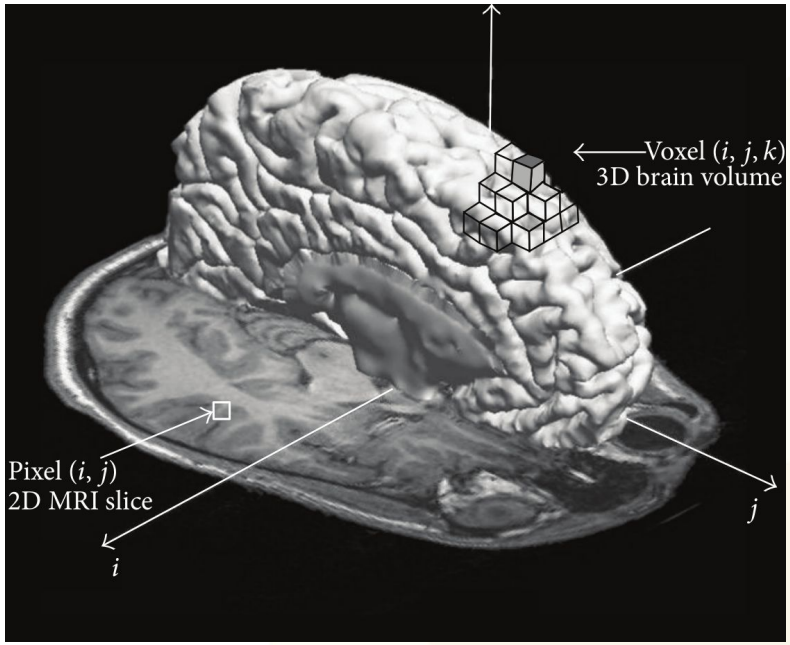
1. MRI machines first create a magnetic field along the axis of the scanner
2. Aligns hydrogen (H) atoms in the body to the magnetic field
3. MRI emits magnetic pulse that knocks H atoms out of alignment
4. Time it takes for H atoms to regain alignment depends on biological tissue
5. MRI image is based on this realignment time and this reflects type of tissue
    a. Realignment speed : WM (bright) > GM (medium) > CSF (dark)



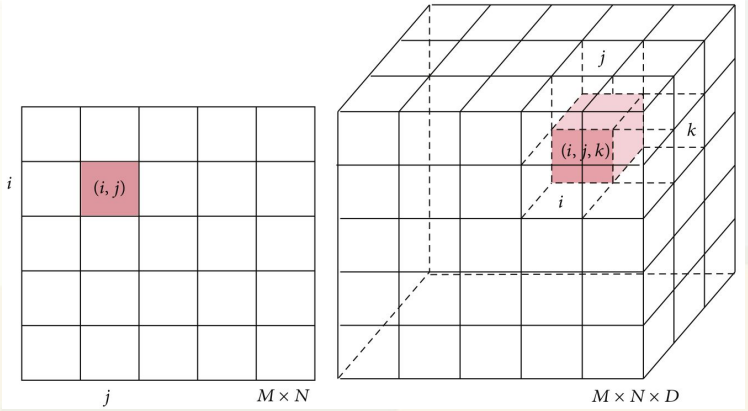


neuro

McGill | Centre universitaire de santé McGill | McGill University Health Centre



Pixel (2D image) Voxel (3D volume)

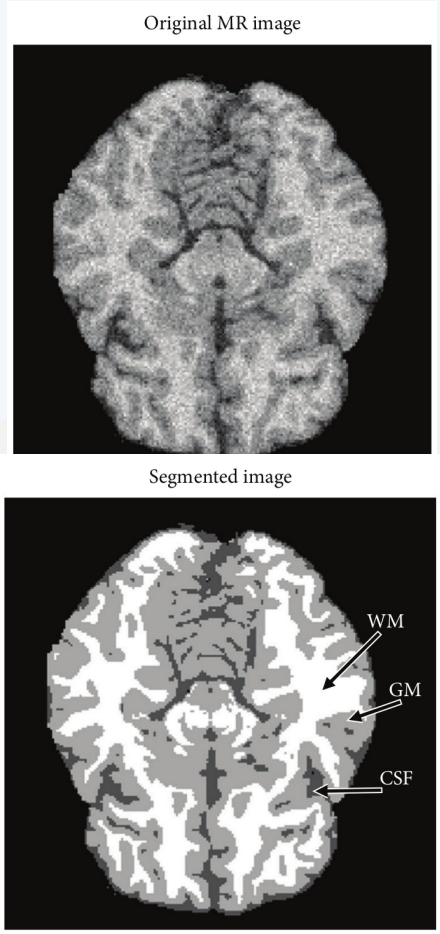


neuro

McGill | Centre universitaire de santé McGill | McGill University Health Centre

MRI Segmentation

- Lots of ways to analyze MRI
  - Brain size
  - Cortical thickness
  - GM/WM intensity ratio



- Segmenting MRI very useful
  - Segmenting into GM and WM is common processing step
  - Helps to quantify brain metrics measured in these regions

- ML can be used to perform segmentation!

neuro

McGill | Centre universitaire de santé McGill | McGill University Health Centre

Outline

1. Brain imaging and neuroanatomy
2. Machine Learning
3. Deep Learning
4. Keras Example

Outline

1. Brain imaging and neuroanatomy
2. Machine Learning
3. Deep Learning
4. Keras Example

Introduction to Deep Learning

- Layers of artificial neurons learn increasingly abstract representations of your data
- Can capture very complex interactions between features using non-linear activations

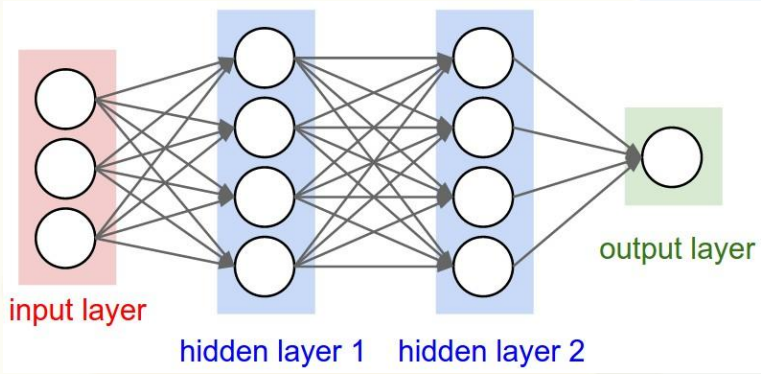


Image: http://cs231n.github.io/convolutional-networks/

neuro

McGill | Centre universitaire de santé McGill | McGill University Health Centre

Components of neural networks

- Each layer has a set off artificial neurons/nodes
- Traditionally activated with rectified linear units (i.e. ReLU)
- Regularization done through dropout

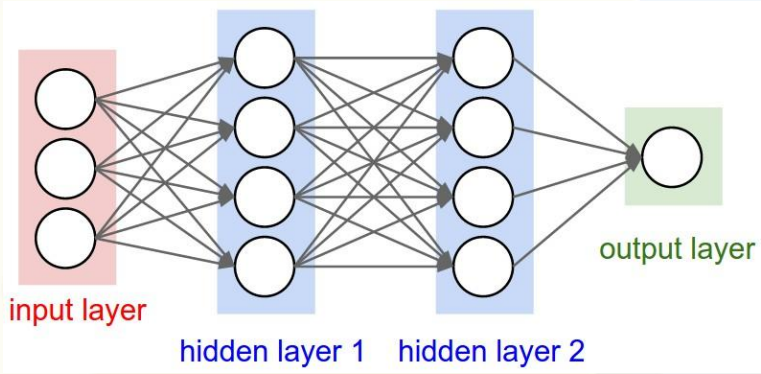


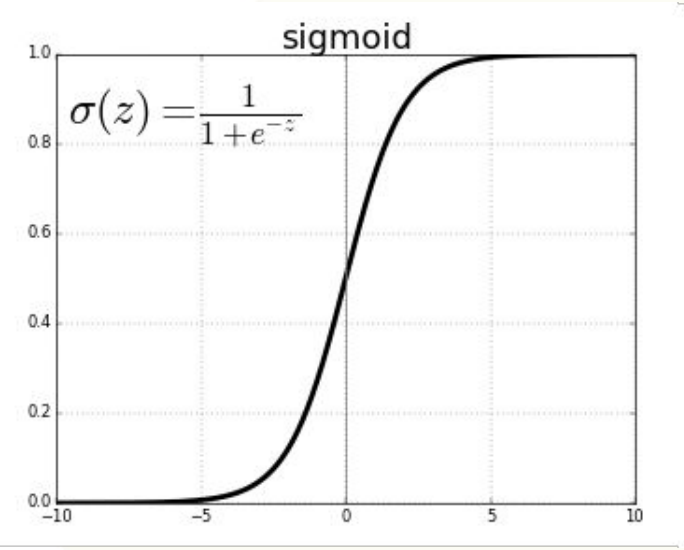
Image: http://cs231n.github.io/convolutional-networks/

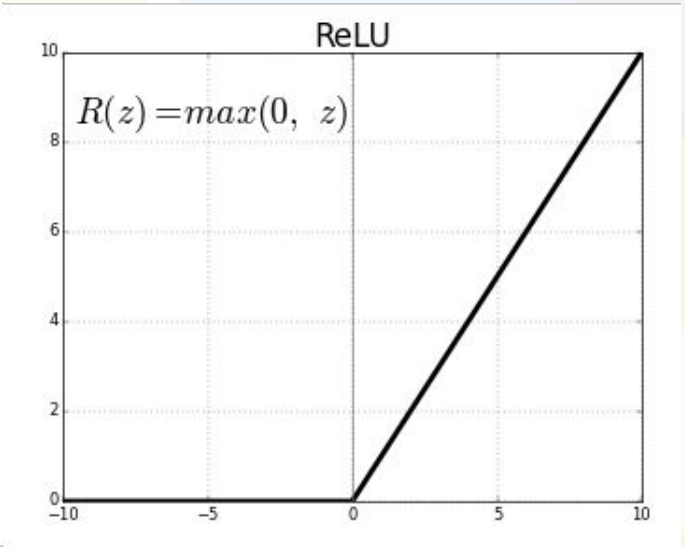
neuro

McGill | Centre universitaire de santé McGill | McGill University Health Centre

Components of neural networks

- Each layer has a set off artificial neurons/nodes
- Traditionally activated with rectified linear units (i.e. ReLU)
- Regularization done through dropout





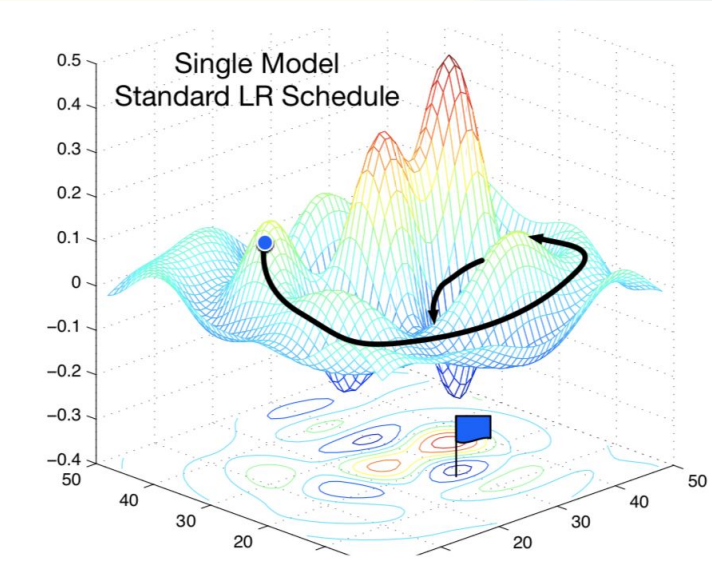
https://cdn-images-1.medium.com/max/1600/1*XxxiA0jJvPrHEJHD4z893g.png

neuro

McGill | Centre universitaire de santé McGill | McGill University Health Centre

Training neural networks

- Stochastic gradient descent
  - Pick a training sample
  - Feed it through the network
  - Determine the error of said prediction
  - Compute the share of correction for all hidden nodes and update the weights
- Many algorithms for updating the weights such as:
  - Gradient Descent
  - RMSProp
  - Adam



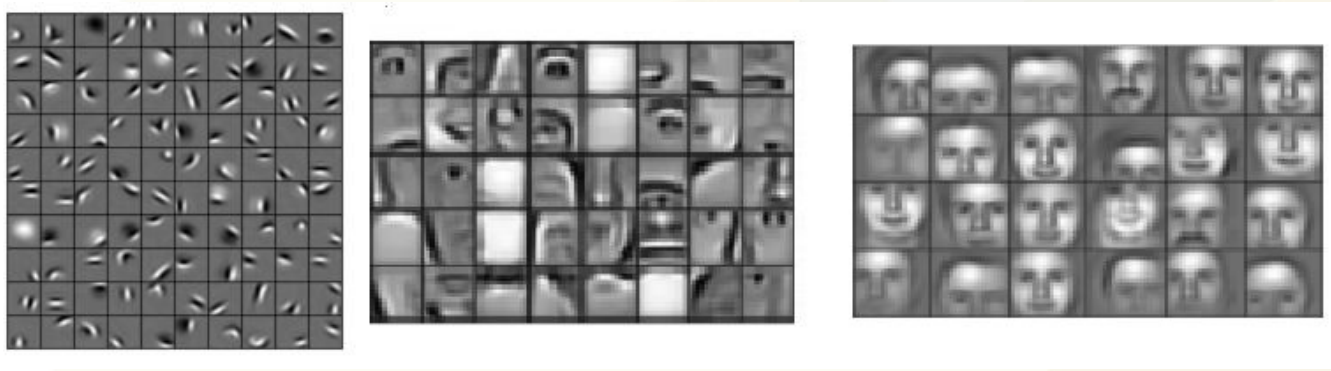
https://cdn-images-1.medium.com/max/2000/1*T5WWecP_EaQWk1yDX15h_w.png

neuro

McGill | Centre universitaire de santé McGill | McGill University Health Centre

Convolutional neural networks

- Typically used in imaging data
- Many feature filters scan over the image to pick out specific features (e.g. edges, curves, eyes, nose, face)
- These filters learn to recognize features that make the performance of the prediction increase

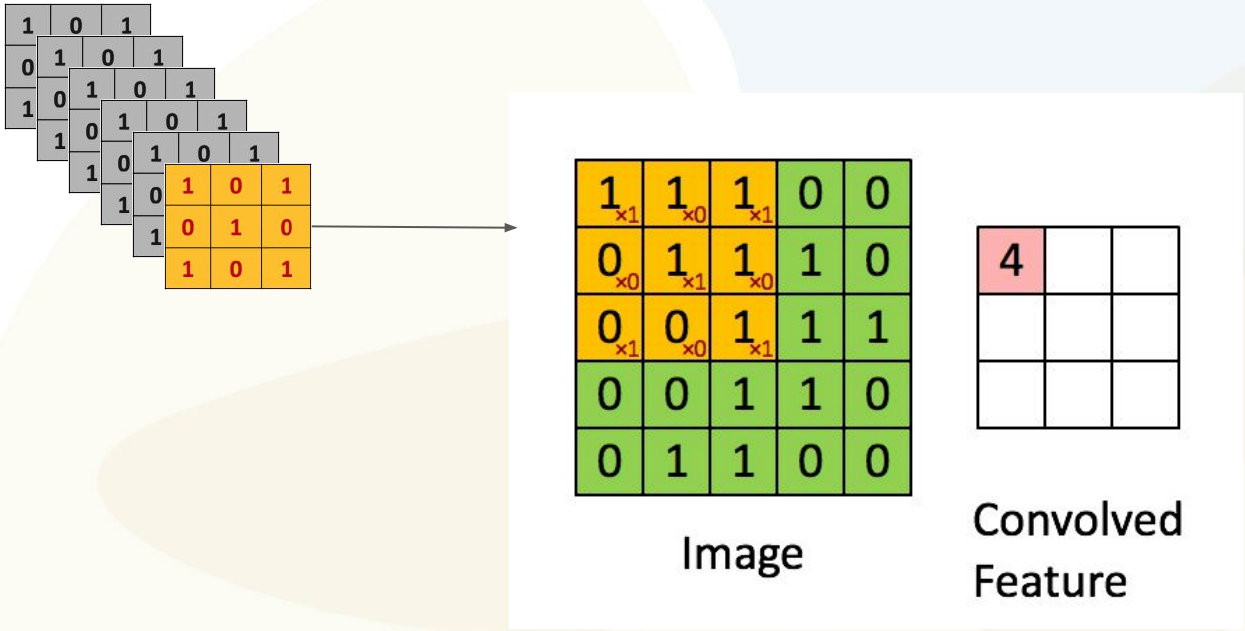


https://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-core-concepts/

neuro

McGill | Centre universitaire de santé McGill | McGill University Health Centre

Learning features through kernels

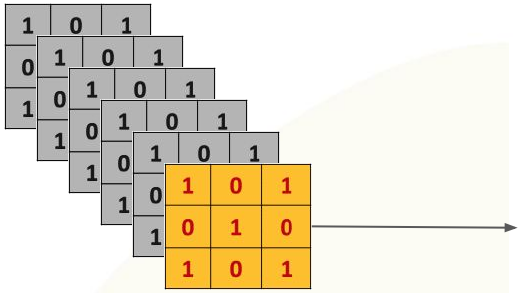


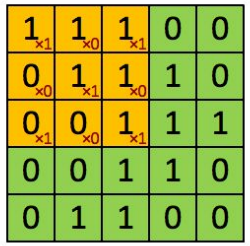
http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

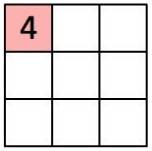
neuro

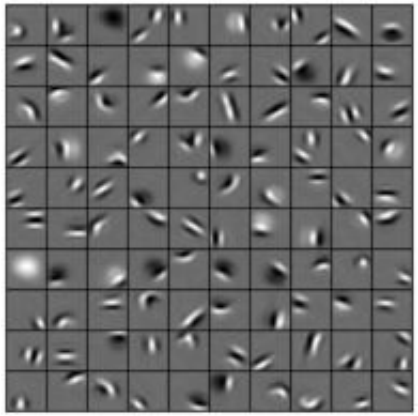
McGill | Centre universitaire de santé McGill | McGill University Health Centre

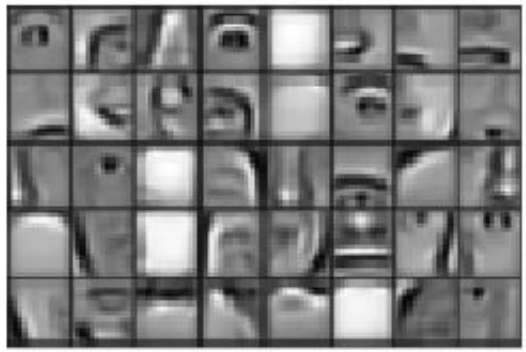
Learning features through kernels









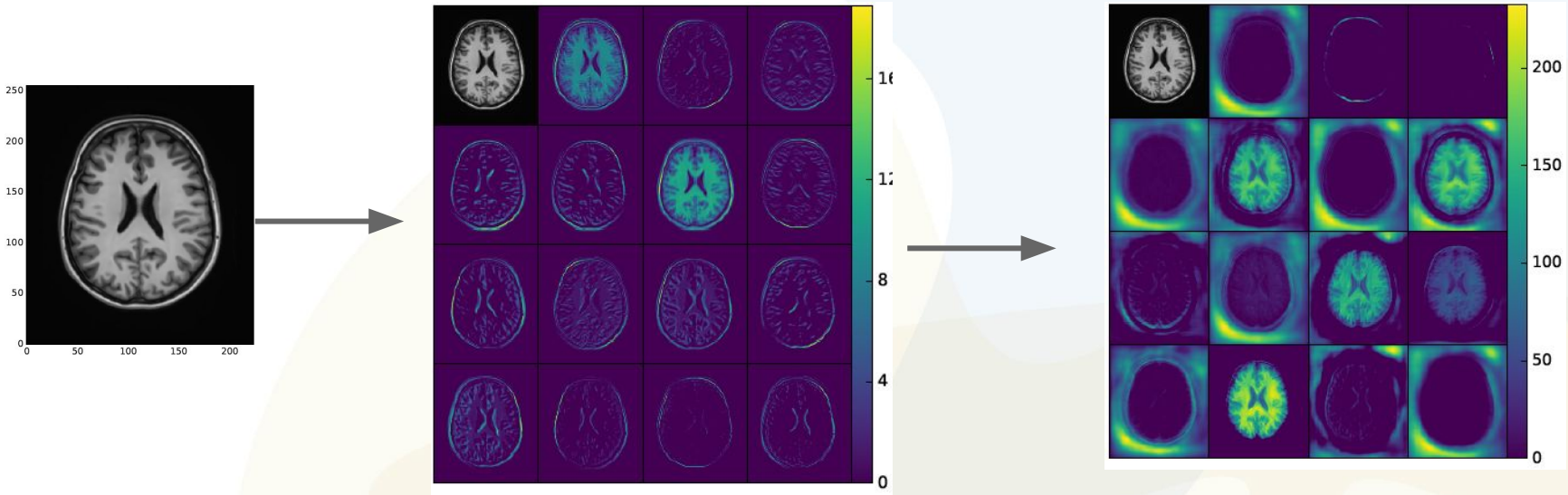


http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

neuro

McGill | Centre universitaire de santé McGill | McGill University Health Centre

Learning brain segmentation



Fratila, R., Fonov, V., Collins, L.D., Arnold, D.L., Brown, R. (2017). DeepDiscovery: Rapid and Efficient Deep Learning for NeuroImaging. NeuroImage (submitted).

neuro

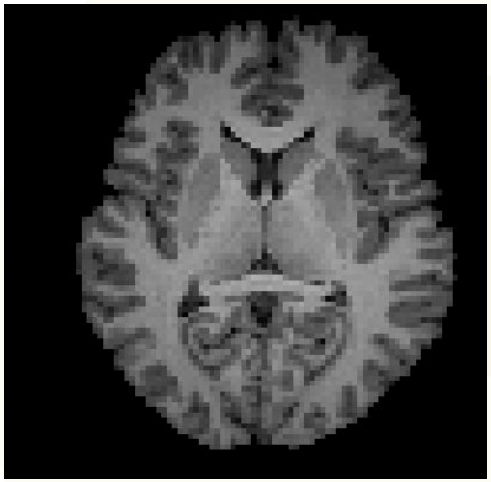
McGill | Centre universitaire de santé McGill | McGill University Health Centre

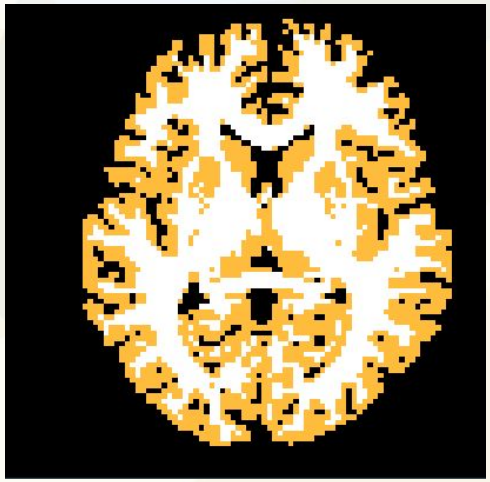
Outline

1. Brain imaging and neuroanatomy
2. Machine Learning
3. Deep Learning
4. Keras Example

1000 Functional Connectomes Project

- 261 MRI (skull stripped)
  - http://fcon_1000.projects.nitrc.org/fcpClassic/FcpTable.html
- Data stored in MINC (.mnc)
  - MINC is a medical imaging format based on HDF5
- GM/WM segmentation with FSL-5.0-fast





neuro

McGill | Centre universitaire de santé McGill | McGill University Health Centre

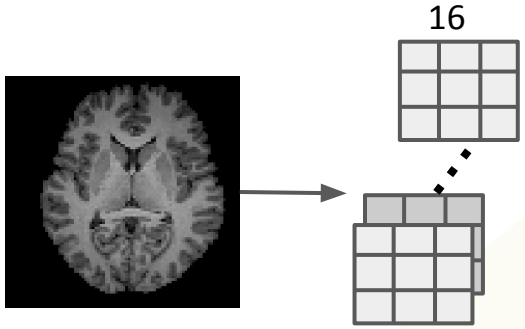
Build a Conv Net with Keras

```
#Setup the input (IN) based on image dimensions <image_dim>
IN = Input(shape=(image_dim[1], image_dim[2],1))

#Create 4 convolutional layers with 3x3 kernels and 16, 32, 64, 128 kernels per layer
CONV1 = Conv2D(16,  kernel_size=[3,3],activation='relu',padding='same')(IN)
CONV2 = Conv2D(32,  kernel_size=[3,3],activation='relu',padding='same')(CONV1)
CONV3 = Conv2D(64,  kernel_size=[3,3],activation='relu',padding='same')(CONV2)
CONV4 = Conv2D(128, kernel_size=[3,3],activation='relu',padding='same')(CONV3)

#Setup output layer (OUT)
OUT = Conv2D(1, kernel_size=1,  padding='same', activation='softmax')(CONV4)

#Create model
model = keras.models.Model(inputs=[IN], outputs=OUT)
```

neuro

McGill | Centre universitaire de santé McGill | McGill University Health Centre

Build a Conv Net with Keras

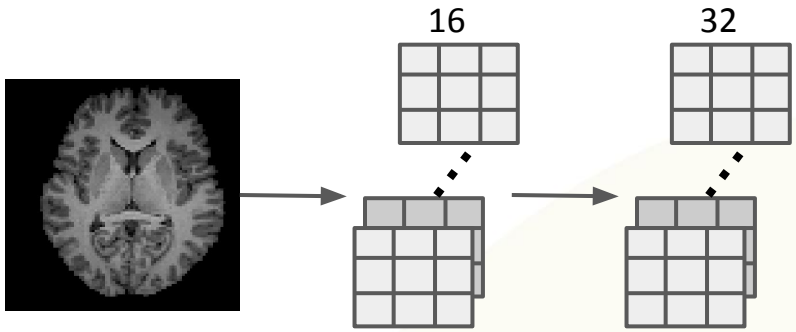



```
#Setup the input (IN) based on image dimensions <image_dim>
IN = Input(shape=(image_dim[1], image_dim[2],1))

#Create 4 convolutional layers with 3x3 kernels and 16, 32, 64, 128 kernels per layer
CONV1 = Conv2D(16,
kernel_size=[3,3],activation= 'relu',padding='same')(IN)
CONV2 = Conv2D(32,  kernel_size=[3,3],activation='relu',padding='same')(CONV1)
CONV3 = Conv2D(64,  kernel_size=[3,3],activation='relu',padding='same')(CONV2)
CONV4 = Conv2D(128, kernel_size=[3,3],activation='relu',padding='same')(CONV3)

#Setup output layer (OUT)
OUT = Conv2D(3, kernel_size=1,  padding='same', activation='softmax')(CONV4)

#Create model
model = keras.models.Model(inputs=[IN], outputs=OUT)
```

neuro

McGill | Centre universitaire de santé McGill | McGill University Health Centre

Build a Conv Net with Keras

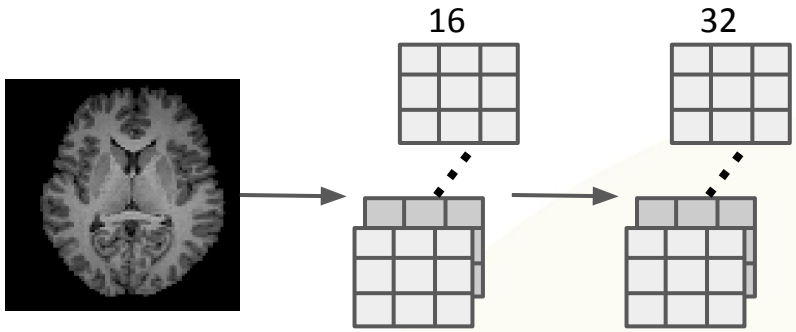



```
#Setup the input (IN) based on image dimensions <image_dim>
IN = Input(shape=(image_dim[1], image_dim[2],1))

#Create 4 convolutional layers with 3x3 kernels and 16, 32, 64, 128 kernels per layer
CONV1 = Conv2D(16,  kernel_size=[3,3],activation='relu',padding='same')(IN)
CONV2 = Conv2D(32,
kernel_size=[3,3],activation= 'relu',padding='same')(CONV1)
CONV3 = Conv2D(64,  kernel_size=[3,3],activation='relu',padding='same')(CONV2)
CONV4 = Conv2D(128, kernel_size=[3,3],activation='relu',padding='same')(CONV3)

#Setup output layer (OUT)
OUT = Conv2D(3, kernel_size=1,  padding='same', activation='softmax')(CONV4)

#Create model
model = keras.models.Model(inputs=[IN], outputs=OUT)
```

neuro

McGill | Centre universitaire de santé McGill | McGill University Health Centre

Build a Conv Net with Keras

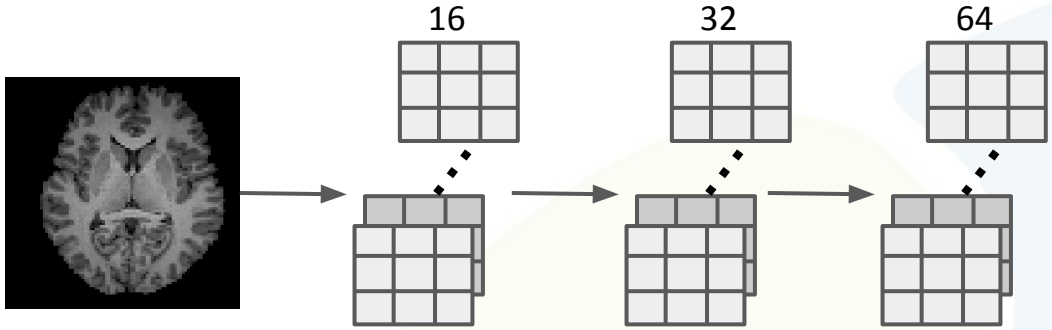



```
#Setup the input (IN) based on image dimensions <image_dim>
IN = Input(shape=(image_dim[1], image_dim[2],1))

#Create 4 convolutional layers with 3x3 kernels and 16, 32, 64, 128 kernels per layer
CONV1 = Conv2D(16,  kernel_size=[3,3],activation='relu',padding='same')(IN)
CONV2 = Conv2D(32,
kernel_size=[3,3],activation= 'relu',padding='same')(CONV1)
CONV3 = Conv2D(64,  kernel_size=[3,3],activation='relu',padding='same')(CONV2)
CONV4 = Conv2D(128, kernel_size=[3,3],activation='relu',padding='same')(CONV3)

#Setup output layer (OUT)
OUT = Conv2D(3, kernel_size=1,  padding='same', activation='softmax')(CONV4)

#Create model
model = keras.models.Model(inputs=[IN], outputs=OUT)
```

neuro

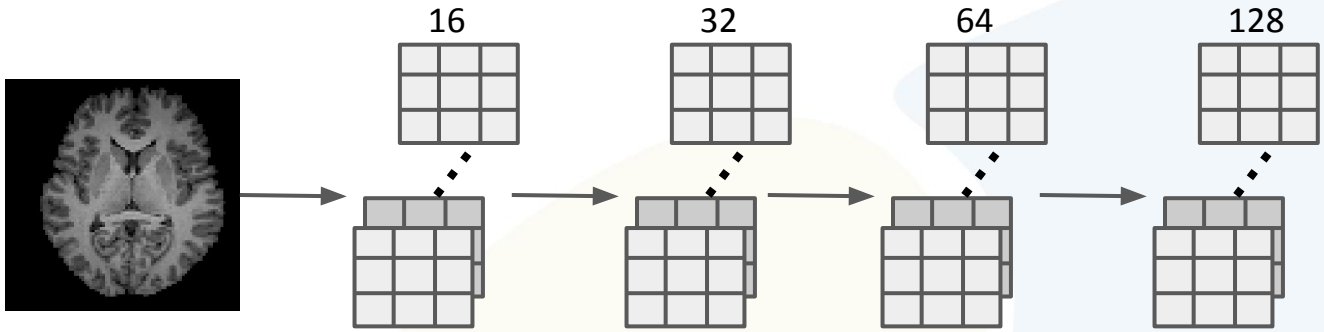
McGill | Centre universitaire de santé McGill | McGill University Health Centre

Build a Conv Net with Keras




```
#Setup the input (IN) based on image dimensions <image_dim>
IN = Input(shape=(image_dim[1], image_dim[2],1))

#Create 4 convolutional layers with 3x3 kernels and 16, 32, 64, 128 kernels per layer
CONV1 = Conv2D(16,  kernel_size=[3,3],activation='relu',padding='same')(IN)
CONV2 = Conv2D(32,  kernel_size=[3,3],activation='relu',padding='same')(CONV1)
CONV3 = Conv2D(64, kernel_size=[3,3],activation= 'relu',padding='same')(CONV2)
CONV4 = Conv2D(128, kernel_size=[3,3],activation='relu',padding='same')(CONV3)

#Setup output layer (OUT)
OUT = Conv2D(3, kernel_size=1,  padding='same', activation='softmax')(CONV4)

#Create model
model = keras.models.Model(inputs=[IN], outputs=OUT)
```

Build a Conv Net with Keras

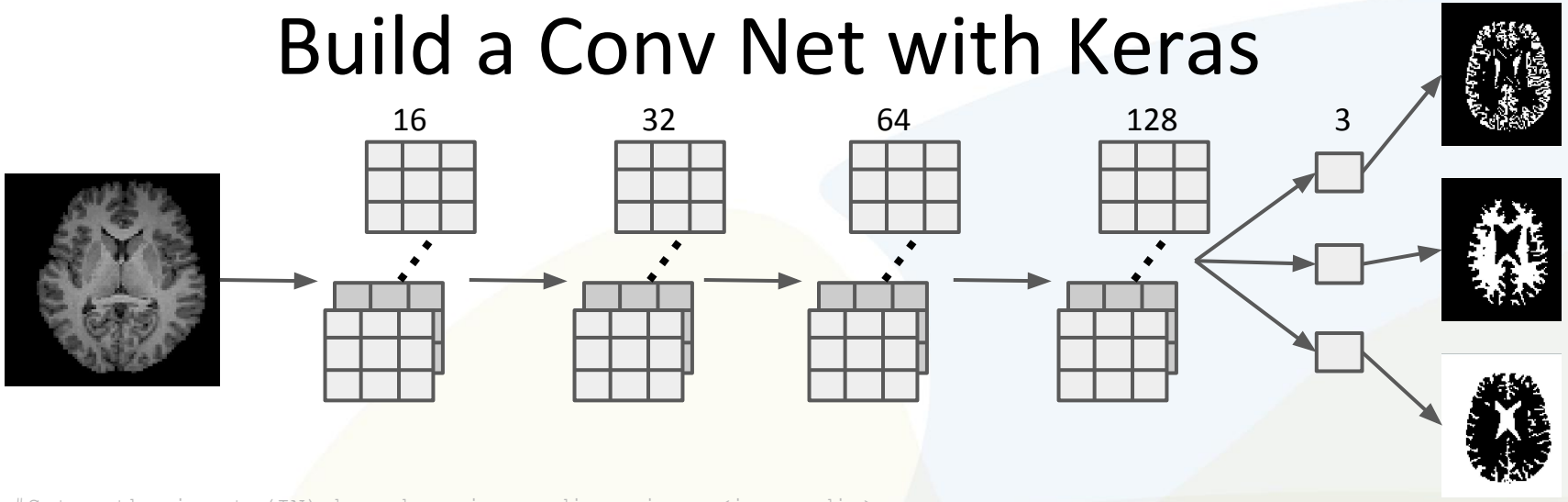



```
#Setup the input (IN) based on image dimensions <image_dim>
IN = Input(shape=(image_dim[1], image_dim[2],1))

#Create 4 convolutional layers with 3x3 kernels and 16, 32, 64, 128 kernels per layer
CONV1 = Conv2D(16,  kernel_size=[3,3],activation='relu',padding='same')(IN)
CONV2 = Conv2D(32,  kernel_size=[3,3],activation='relu',padding='same')(CONV1)
CONV3 = Conv2D(64,  kernel_size=[3,3],activation='relu',padding='same')(CONV2)
CONV4 = Conv2D(128,
kernel_size=[3,3],activation='relu',padding='same')(CONV3)

#Setup output layer (OUT)
OUT = Conv2D(3, kernel_size=1,  padding='same', activation='softmax')(CONV4)

#Create model
model = keras.models.Model(inputs=[IN], outputs=OUT)
```

neuro

McGill | Centre universitaire de santé McGill | McGill University Health Centre

Build a Conv Net with Keras




```
#Setup the input (IN) based on image dimensions <image_dim>
IN = Input(shape=(image_dim[1], image_dim[2],1))

#Create 4 convolutional layers with 3x3 kernels and 16, 32, 64, 128 kernels per layer
CONV1 = Conv2D(16,  kernel_size=[3,3],activation='relu',padding='same')(IN)
CONV2 = Conv2D(32,  kernel_size=[3,3],activation='relu',padding='same')(CONV1)
CONV3 = Conv2D(64,  kernel_size=[3,3],activation='relu',padding='same')(CONV2)
CONV4 = Conv2D(128, kernel_size=[3,3],activation='relu',padding='same')(CONV3)

#Setup output layer (OUT)
OUT = Conv2D(3, kernel_size=1,  padding='same',
activation='softmax')(CONV4)

#Create model
model = keras.models.Model(inputs=[IN], outputs=OUT)
```

neuro

McGill | Centre universitaire de santé McGill | McGill University Health Centre

Setup & Fit Model

make_and_run_model.py

```
def compile_and_run(X_train, Y_train, X_validate, Y_validate, epochs):
    #setup optimizer
    ada = keras.optimizers.Adam(0.0001)

    #setup define loss function
    loss function = 'categorical_crossentropy'

    #compile the model
    model.compile(loss = loss_function, optimizer=ada,metrics=[categorical_accuracy] )

    #fit model
    model.fit([X_train],Y_train,batch_size,validation_data=([X_validate],Y_validate),epochs=epo
chs])

    #save model
    model.save(model_name)
```

Base Model

models/neurotech_models.py

```
def base_model(image_dim, nK, kernel_size, drop_out):
     # nK = number of kernels per layer
     # kernel_size = size of the kernels
     # dropout = dropout rate for each layer

     #Setup the input (IN) and output (OUT) layers based on image dimensions
     IN = OUT = Input(shape=(image_dim[1], image_dim[2],1))

     n_layers=int(len(nK)) # number of layer
     kDim=[kernel_size] * n_layers #list of kernel size equal in length to n_layers

     for i in range(n_layers): # for each layer...
          OUT=Conv2D(nK[i],kernel_size=[kDim[i],kDim[i]],activation='relu',padding='same')(OUT)
          OUT = Dropout(drop_out)(OUT)

     OUT = Conv2D(1, kernel_size=1,  padding='same', activation='sigmoid')(OUT)
     model = keras.models.Model(inputs=[IN], outputs=OUT)
     return(model)
```

model_0_0

models/neurotech_models.py

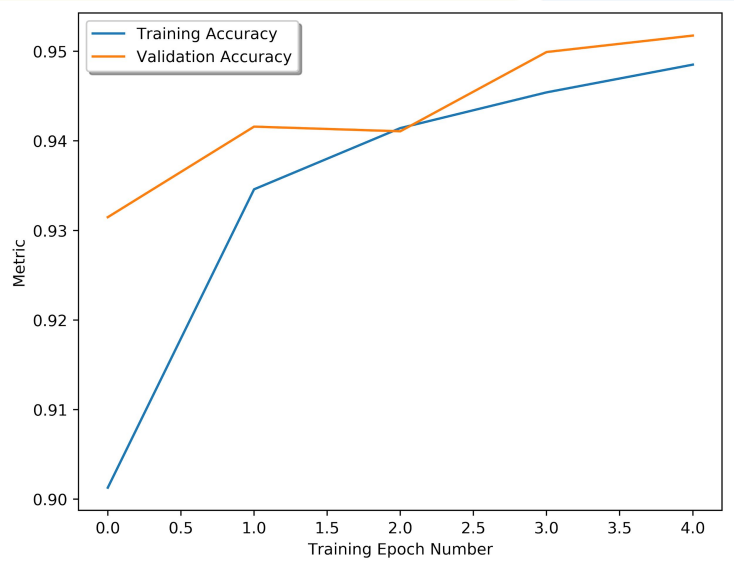
```
def model_0_0(image_dim):
      # Create a convolutional network with
      # [3,3] x 16
      # [3,3] x 16
      # [3,3] x 32
      # [3,3] x 32
      # [3,3] x 64
      # [3,3] x 64
      nK=[16,16,32,32,64,64]
      kernel_size = 3
      drop_out=0
      return base_model( image_dim, nK, kernel_size, drop_out)
```

How to run a model with *minc_keras.py*

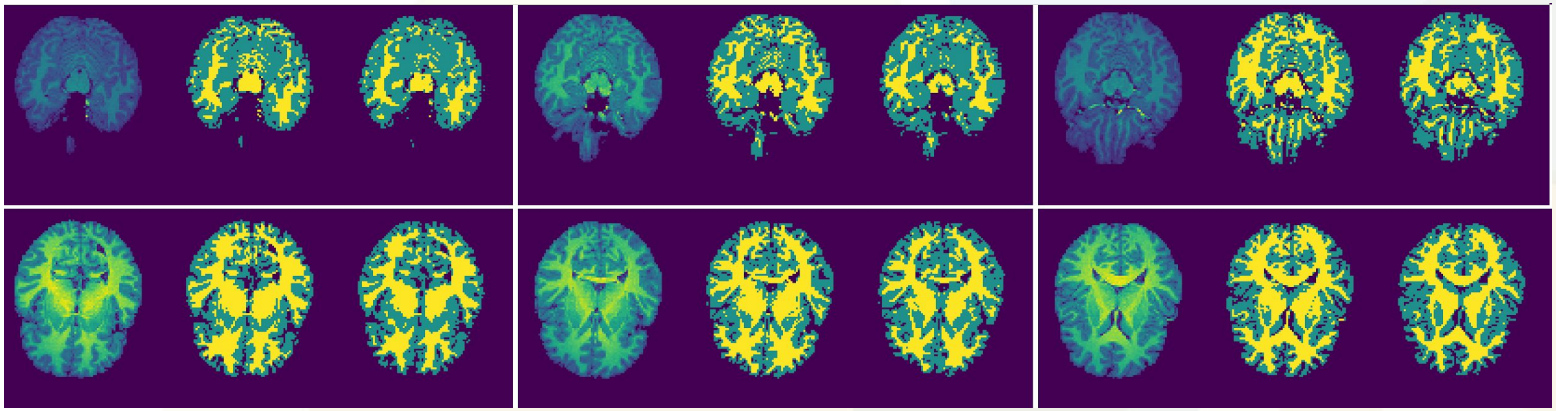
```
python3 minc_keras/minc_keras.py
          --source output/                # source dir where input data is stored
          --target .                      # output dir where results will be placed
          --epochs 5                      # number of epochs to use to fit model
          --model-type "model_0_0”        # name of model you want to use
          --input-str "*T1w_anat*"        # string that identifies the input MRI images
          --label-str "*seg*"             # string that identifies the labeled GM/WM images
          --predict 1                     # comma-separated list of subjects for prediction
          --ratios 0.7 0.15               # ratio to use in training/validation/test splits
          --activation-hidden "relu"      # activation function for hidden layer
          --activation-output "softmax" # activation function for output layer
```

Results for model_0_0

- Parameters = 71,987
- Test Accuracy= 0.951



Results for model_0_0



model_1_0

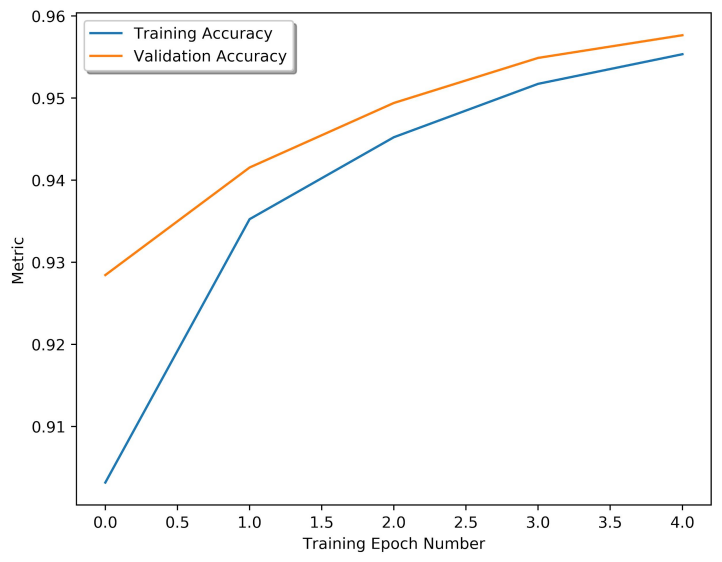
models/neurotech_models.py

```
def model_1_0(image_dim):
      # Create a convolutional network with
      # [3,3] x 16
      # [3,3] x 16
      # [3,3] x 32
      # [3,3] x 32
      # [3,3] x 32
      # [3,3] x 64
      # [3,3] x 64
      # [3,3] x 64
      # [3,3] x 128
      nK=[16,16,32,32,32,64,64,64,128]
      kernel_size = 3
      drop_out=0
      return base_model( image_dim, nK, kernel_size, drop_out)
```

```
python3 minc_keras/minc_keras.py --source output/ --target . --epochs 5 --model-type model_1_0"
--input-str "*T1w_anat*" --label-str "*seg*" --predict 1 --ratios .1 .1
```

Results for model_1_0

- Parameters = 192,211
  - model_0_0: 71,987
- Test Accuracy = 0.957
  - model_0_0: 0.951



neuro

McGill | Centre universitaire de santé McGill | McGill University Health Centre

model_2_0

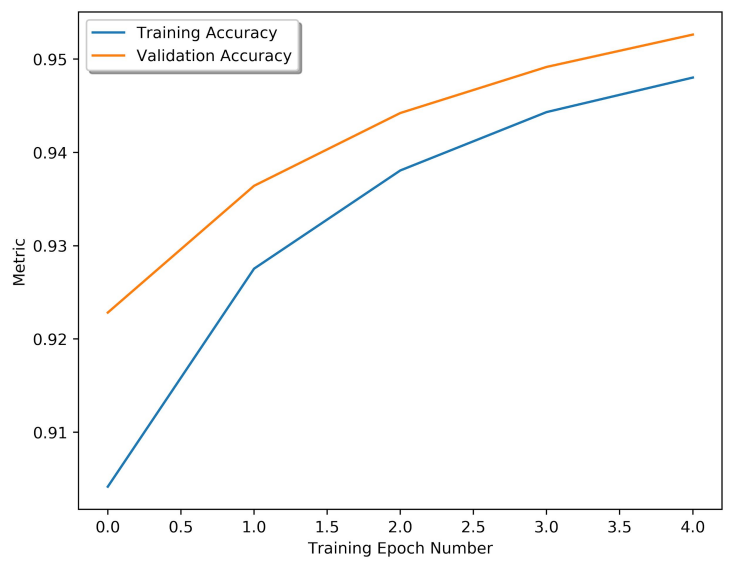
models/neurotech_models.py

```
def model_2_0(image_dim):
      # Create a convolutional network with
      # [5,5] x 16
      # [5,5] x 16
      # [5,5] x 32
      # [5,5] x 32
      # [5,5] x 64
      # [5,5] x 64
      nK=[16,16,32,32,64,64]
      kernel_size = 5
      drop_out=0
      return base_model( image_dim, nK, kernel_size, drop_out)
```

```
python3 minc_keras/minc_keras.py --source output/ --target . --epochs 5 --model-type model_2_0"
--input-str "*T1w_anat*" --label-str "*seg*" --predict 1 --ratios .1 .1
```

Results for model_2_0

- Parameters = 199,219
  - model_0_0: 71,987
- Metric Accuracy = 0.952
  - model_0_0: 0.951



neuro

McGill | Centre universitaire de santé McGill | McGill University Health Centre

model_4_0

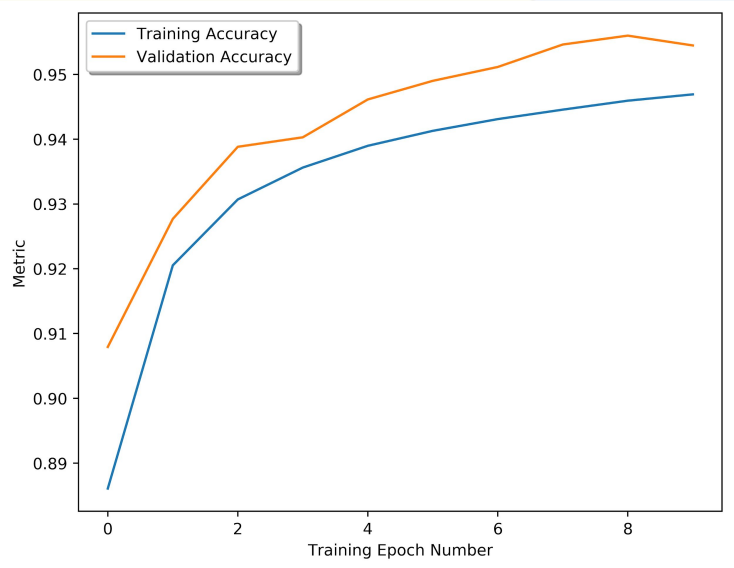
models/neurotech_models.py

```
def model_1_0(image_dim):
      # Create a convolutional network with
      # [5,5] x 16
      # [5,5] x 16
      # [5,5] x 32
      # [5,5] x 32
      # [5,5] x 64
      # [5,5] x 64
      nK=[16,16,32,32,64,64]
      kernel_size = 5
      drop_out=0.25
      return base_model( image_dim, nK, kernel_size, drop_out)
```

```
python3 minc_keras/minc_keras.py --source output/ --target . --epochs 5 --model-type model_4_0"
--input-str "*T1w_anat*" --label-str "*seg*" --predict 1 --ratios .1 .1
```

Results for model_4_0

- Parameters = 532,435
  - model_0_0: 71,987
- Test Accuracy: 0.954
  - model_0_0: 0.951



Free coding

1. Can you find a model that does better than 95% without overfitting?
   a. What happens when you change the activation functions?

2. You can change the training/validation/test ratio to increase samples for training
   a. e.g., --train 0.3 0.3 (30% training, 30% validation, 60% split)
   b. a large training set will slow down the training, but reduce overfitting

3. Ask volunteers and organizers for help!

Acknowledgments

- Paul Lemaitre, PhD
  - Slides + writing code for minc_keras