General Approaches of Histopathological Image Classification using Convolutional Neural Networks

Jerome Cheng M.D.
Disclosures

none
Outline

Background on Convolutional Neural Networks
  description
  applications
  layers
Image classification strategies
  train from scratch
  transfer learning
Tools
  Orange
  Keras
Convolutional Neural Networks

class of deep, feed forward artificial neural networks usually composed of convolution, pooling, fully connected layers

most commonly applied in pathology for image classification and segmentation tasks

Inspired by Dr. Hubel and Dr. Wiesel’s work on the cat visual system in the 50’s and 60’s that showed a hierarchical arrangement of neurons and some neurons were excited by lines in a particular orientation in a specific location
Artificial Neural Network Example

\[ a_1w_1 + a_2w_2 + a_3w_3 + \text{bias} \rightarrow \text{Relu} \rightarrow \text{Output} \]
Images

conv2d_1: Conv2D

conv2d_2: Conv2D

max_pooling2d_1: MaxPooling2D

conv2d_3: Conv2D

max_pooling2d_2: MaxPooling2D

flatten_1: Flatten

dense_1: Dense

dense_2: Dense

Fully Connected Layer

Prediction
Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position

Kunihiko Fukushima

NHK Broadcasting Science Research Laboratories, Kinuta, Setagaya, Tokyo, Japan

Abstract. A neural network model for a mechanism of visual pattern recognition is proposed in this paper. The network is self-organized by "learning without a teacher", and acquires an ability to recognize stimulus patterns based on the geometrical similarity (Gestalt) of their shapes without affected by their positions. This reveal it only by conventional physiological experiments. So, we take a slightly different approach to this problem. If we could make a neural network model which has the same capability for pattern recognition as a human being, it would give us a powerful clue to the understanding of the neural mechanism in the
Acceleration of CNN Progress and Adaptation

Faster PCs and GPUs

Publicly available image datasets

Open source CNN deep learning frameworks
ImageNet

Online database containing more than 14 million images belonging to 20,000 categories

Large training dataset is important in deep learning
  - Several pre-trained CNN models were trained on ImageNet

Accelerated the progress of CNN research

Hosted yearly competitions

Many publicly available pre-trained CNN models were trained on ImageNet

http://www.image-net.org/
Layers

Convolutional
Pooling
Fully connected
Activation layers
Dropout layer
Regularization layers

Low level features

High level features
Convolutional Neural Network Layers

Image Data

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>4</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Filter

1  -1  1
0   0  0
-1  0  0

Convolution

1x1 + 2x(-1) + 4x1 + 2x0 + 0x0 + 2x0 + 1x(-1) + 0x0 + 1x0 = 2

Pooling

2  3
3  1
Training

Loss function - measure the difference between the predicted value and the expected value

Weights are adjusted in the convolution and fully connected (dense) layers to minimize the loss function
Pre-trained CNN architectures

VGG-16
VGG-19
Inception
XCeption
MobileNet
Applications of CNN in Pathology

Image Classification

Image Segmentation

Object localization and classification
Image Classification

Image Segmentation

Ground Truth

Prediction
Classification Strategies (CNN)

Train from scratch

- Weights from all layers are updated

Transfer learning

- Freeze lower layers
  - Only weights from some layers are updated
- Combine pre-classification layer output with machine learning algorithms
  - None of the weights are updated
Transfer Learning - Freezing Layers

Weights are updated only on selected layers

Information learned from a previous dataset (ImageNet) is preserved

Speeds up training
Transfer Learning - Combine Output with Another Machine Learning Algorithm

The final classification layer of the pre-trained network should not be included.

Ex. an image processed through VGG 16 = 4096 features

4096 features + image label can be trained using various machine learning algorithms like Random Forest, logistic regression, and SVM.

None of the weight parameters are updated during training.
Steps – Classifying Images with CNN

Scan slides

Create and label tiles - ex. 224x224 or 299x299 pixels

Training and Validation set

Augmentation

Choose or Create a Model

Model Training

Prediction
Augmentation

Rotation
Shear
Channel Shift
Scaling
Height shift
Width shift

Download the latest version for Windows

**Download Orange 3.20**

Miniconda installer (Default)
Orange3-3.20.1-Miniconda-x86_64.exe (64 bit)
Installs Miniconda and Orange. Can be used without administrator privileges.
Please report any problems to our Issue Tracker.

Classic Installer
Orange3-3.20.1-Python36-win32.exe
Installs Orange along with Python and all required libraries (Python, NumPy, SciPy, SciKit Learn, PyQt, ...)

Orange (Biolab) - https://orange.biolab.si
Data preparation

Create image tiles from the digital slide or image (224 x 224 for VGG 16)

Create a folder with subfolders for each image Class

Ex. Training Set
- Cancer
- Benign
Orange3-Associate

Orange add-on for enumerating frequent itemsets and association rules mining.

Documentation: [http://orange3-associate.readthedocs.org/](http://orange3-associate.readthedocs.org/)
Orange - loading images
Orange - Image Embedding

Data → Images

Import Images

Image Embedding

Data with 300 instances.
Connected to server.

Settings

Image attribute: [image]
Embedder: [VGG-16]
16-layer image recognition model trained on ImageNet.

Apply Automatically

Cancel

Embeddings → Data

Data → Images

Test & Score

Model → Predictions

Learner
Orange - Hardware (image embeddings)

Cluster of 8 machines

6 machines - 2 processors with 4 cores each (Intel Xenon CPU), 32 GB RAM

2 machines - 2 processors with 8 cores (Intel Xeon e5-2660), 157 GB RAM

Embedders are automatically assigned to available CPU cores

Soon: switch to an improved backend with a combination of CPUs and GPUs, with the addition of 3 TitanX GPUs (12 GB ram each)
Orange - data visualization

Dimensional reduction + scatter plot

Hierarchical clustering
Keras

High level neural network library that works on top of TensorFlow, CNTK, or Theano

Written in Python

CNN models can be built and trained with very little code

Works with CPU and GPUs
import numpy as np
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D
from keras.optimizers import SGD

x_train = np.random.random(((100, 100, 100, 3)))
y_train = keras.utils.to_categorical(np.random.randint(10, size=(100, 1)), num_classes=10)
x_test = np.random.random(((20, 100, 100, 3)))
y_test = keras.utils.to_categorical(np.random.randint(10, size=(20, 1)), num_classes=10)

model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(100, 100, 3)))
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))

sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd)

model.fit(x_train, y_train, batch_size=32, epochs=10)
score = model.evaluate(x_test, y_test, batch_size=32)
1219/1253 [==================================] - ETA: 31s - loss: 0.1287 - acc: 0.9553
1220/1253 [==================================] - ETA: 30s - loss: 0.1286 - acc: 0.9553
1221/1253 [==================================] - ETA: 29s - loss: 0.1286 - acc: 0.9553
1222/1253 [==================================] - ETA: 28s - loss: 0.1287 - acc: 0.9553
1223/1253 [==================================] - ETA: 27s - loss: 0.1286 - acc: 0.9553
1224/1253 [==================================] - ETA: 26s - loss: 0.1288 - acc: 0.9553
1225/1253 [==================================] - ETA: 25s - loss: 0.1287 - acc: 0.9553
1226/1253 [==================================] - ETA: 24s - loss: 0.1287 - acc: 0.9553
1227/1253 [==================================] - ETA: 23s - loss: 0.1286 - acc: 0.9554
1228/1253 [==================================] - ETA: 22s - loss: 0.1286 - acc: 0.9554
1229/1253 [==================================] - ETA: 21s - loss: 0.1286 - acc: 0.9554
1230/1253 [==================================] - ETA: 20s - loss: 0.1286 - acc: 0.9554
1231/1253 [==================================] - ETA: 19s - loss: 0.1286 - acc: 0.9554
1232/1253 [==================================] - ETA: 18s - loss: 0.1286 - acc: 0.9554
1233/1253 [==================================] - ETA: 17s - loss: 0.1286 - acc: 0.9554
1234/1253 [==================================] - ETA: 16s - loss: 0.1286 - acc: 0.9554
1235/1253 [==================================] - ETA: 15s - loss: 0.1286 - acc: 0.9554
1236/1253 [==================================] - ETA: 14s - loss: 0.1287 - acc: 0.9553
1237/1253 [==================================] - ETA: 13s - loss: 0.1286 - acc: 0.9554
1238/1253 [==================================] - ETA: 12s - loss: 0.1286 - acc: 0.9554
1239/1253 [==================================] - ETA: 11s - loss: 0.1286 - acc: 0.9554
1240/1253 [==================================] - ETA: 10s - loss: 0.1286 - acc: 0.9554
1241/1253 [==================================] - ETA: 9s - loss: 0.1286 - acc: 0.9554
1242/1253 [==================================] - ETA: 8s - loss: 0.1286 - acc: 0.9554
1243/1253 [==================================] - ETA: 7s - loss: 0.1286 - acc: 0.9554
1244/1253 [==================================] - ETA: 6s - loss: 0.1286 - acc: 0.9554
1245/1253 [==================================] - ETA: 5s - loss: 0.1286 - acc: 0.9554
1246/1253 [==================================] - ETA: 4s - loss: 0.1286 - acc: 0.9554
1247/1253 [==================================] - ETA: 3s - loss: 0.1286 - acc: 0.9554
1248/1253 [==================================] - ETA: 2s - loss: 0.1286 - acc: 0.9554
1249/1253 [==================================] - ETA: 1s - loss: 0.1286 - acc: 0.9554
1250/1253 [==================================] - ETA: 0s - loss: 0.1286 - acc: 0.9554
1251/1253 [==================================] - 1314s 1s/step - loss: 0.1287 - acc: 0.9554 - val_loss: 0.1553 - val_acc: 0.9490

Epoch 00002: val_acc improved from 0.93401 to 0.94901, saving model to d:\checkpoint\hpcvg.h5

Epoch 3/1000
Convolutional Neural Networks can be used to classify and segment histopathological images.

Transfer learning applies knowledge learned from a different dataset onto a new image subject matter.

Orange (Biolab) is a GUI based machine learning tool that can be used for transfer learning.

Keras is a high level framework written in Python for deep learning.
References


Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXivpreprint

References


