AI for Medicine Course 1 Week 1 lecture exercises

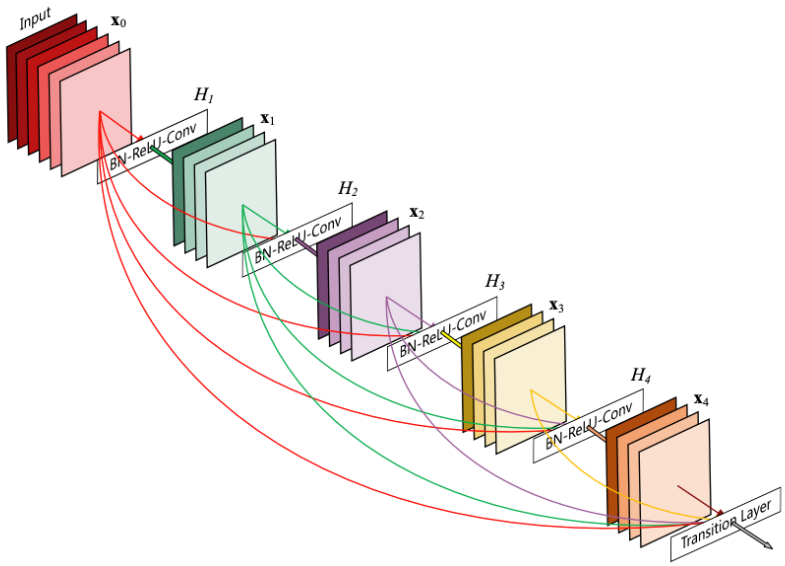
Densenet

In this week's assignment, you'll be using a pre-trained Densenet model for image classification.

Densenet is a convolutional network where each layer is connected to all other layers that are deeper in the network

- The first layer is connected to the 2nd, 3rd, 4th etc.
- The second layer is connected to the 3rd, 4th, 5th etc.

Like this:



For a detailed explanation of Densenet, check out the source of the image above, a paper by Gao Huang et al. 2018 called Densely Connected Convolutional Networks (https://arxiv.org/pdf/1608.06993.pdf).

The cells below are set up to provide an exploration of the Keras densenet implementation that you'll be using in the assignment. Run these cells to gain some insight into the network architecture.

In [1]:
```
# Import Densenet from Keras
from keras.applications.densenet import DenseNet121
from keras.layers import Dense, GlobalAveragePooling2D
from keras.models import Model
from keras import backend as K
```

```
Using TensorFlow backend.
```

For your work in the assignment, you'll be loading a set of pre-trained weights to reduce training time.

In [2]:
```
# Create the base pre-trained model
base_model = DenseNet121(weights='./nih/densenet.hdf5', include_top=F
```

```
WARNING:tensorflow:From /opt/conda/lib/python3.6/site-packages/te
nsorflow_core/python/ops/resource_variable_ops.py:1630: calling B
aseResourceVariable.__init__ (from tensorflow.python.ops.resource
_variable_ops) with constraint is deprecated and will be removed
in a future version.
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
WARNING:tensorflow:From /opt/conda/lib/python3.6/site-packages/ke
ras/backend/tensorflow_backend.py:4070: The name tf.nn.max_pool i
s deprecated. Please use tf.nn.max_pool2d instead.

WARNING:tensorflow:From /opt/conda/lib/python3.6/site-packages/ke
ras/backend/tensorflow_backend.py:4074: The name tf.nn.avg_pool i
s deprecated. Please use tf.nn.avg_pool2d instead.
```

View a summary of the model

In [3]:
```
# Print the model summary
base_model.summary()
```

```
conv4_block9_concat[0][0]

conv4_block10_2_conv[0][0]
__________________________________________________________________________________
__________________________________
conv4_block11_0_bn (BatchNormal (None, None, None, 5 2304
conv4_block10_concat[0][0]
__________________________________________________________________________________
__________________________________
conv4_block11_0_relu (Activatio (None, None, None, 5 0
conv4_block11_0_bn[0][0]
__________________________________________________________________________________
__________________________________
conv4_block11_1_conv (Conv2D)   (None, None, None, 1 73728
conv4_block11_0_relu[0][0]
__________________________________________________________________________________
__________________________________
conv4_block11_1_bn (BatchNormal (None, None, None, 1 512
conv4_block11_1_conv[0][0]
```

In [4]:
```
# Print out the first five layers
layers_l = base_model.layers

print("First 5 layers")
layers_l[0:5]
```

```
First 5 layers
```

Out[4]:
```
[<keras.engine.input_layer.InputLayer at 0x7f7a97aed4e0>,
 <keras.layers.convolutional.ZeroPadding2D at 0x7f7a97aed7b8>,
 <keras.layers.convolutional.Conv2D at 0x7f7a97aed8d0>,
 <keras.layers.normalization.BatchNormalization at 0x7f7a97aedf28>,
 <keras.layers.core.Activation at 0x7f7a97aedf60>]
```

In [5]:
```
# Print out the last five layers
print("Last 5 layers")
layers_l[-6:-1]
```

```
Last 5 layers
```

Out[5]:
```
[<keras.layers.normalization.BatchNormalization at 0x7f7a186c09b0>,
 <keras.layers.core.Activation at 0x7f7a186c0cc0>,
 <keras.layers.convolutional.Conv2D at 0x7f7a186d25c0>,
 <keras.layers.merge.Concatenate at 0x7f7a186797b8>,
 <keras.layers.normalization.BatchNormalization at 0x7f7a186798d0>]
```

In [6]:
```
# Get the convolutional layers and print the first 5
conv2D_layers = [layer for layer in base_model.layers
                 if str(type(layer)).find('Conv2D') > -1]
print("The first five conv2D layers")
conv2D_layers[0:5]
```

```
The first five conv2D layers
```

Out[6]:
```
[<keras.layers.convolutional.Conv2D at 0x7f7a97aed8d0>,
 <keras.layers.convolutional.Conv2D at 0x7f7a94289fd0>,
 <keras.layers.convolutional.Conv2D at 0x7f7a9423dcf8>,
 <keras.layers.convolutional.Conv2D at 0x7f7a941f9dd8>,
 <keras.layers.convolutional.Conv2D at 0x7f7a941b3c88>]
```

In [7]:
```
# Print out the total number of convolutional layers
print(f"There are {len(conv2D_layers)} convolutional layers")
```

```
There are 120 convolutional layers
```

In [8]:
```
# Print the number of channels in the input
print("The input has 3 channels")
base_model.input
```

```
The input has 3 channels
```

Out[8]:
```
<tf.Tensor 'input_1:0' shape=(?, ?, ?, 3) dtype=float32>
```

In [9]:
```
# Print the number of output channels
print("The output has 1024 channels")
x = base_model.output
x
```

```
The output has 1024 channels
```

Out[9]:
```
<tf.Tensor 'relu/Relu:0' shape=(?, ?, ?, 1024) dtype=float32>
```

In [10]:
```
# Add a global spatial average pooling layer
x_pool = GlobalAveragePooling2D()(x)
x_pool
```

Out[10]:
```
<tf.Tensor 'global_average_pooling2d_1/Mean:0' shape=(?, 1024) dtype
=float32>
```

In [11]:
```
# Define a set of five class labels to use as an example
labels = ['Emphysema',
          'Hernia',
          'Mass',
          'Pneumonia',
          'Edema']
n_classes = len(labels)
print(f"In this example, you want your model to identify {n_classes}
```

```
In this example, you want your model to identify 5 classes
```

In [12]:
```
# Add a logistic layer the same size as the number of classes you're
predictions = Dense(n_classes, activation="sigmoid")(x_pool)
print(f"Predictions have {n_classes} units, one for each class")
predictions
```

```
Predictions have 5 units, one for each class
```

Out[12]:
```
<tf.Tensor 'dense_1/Sigmoid:0' shape=(?, 5) dtype=float32>
```

In [13]:
```
# Create an updated model
model = Model(inputs=base_model.input, outputs=predictions)
```

In [14]:
```
# Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy')
# (You'll customize the loss function in the assignment!)
```

This has been a brief exploration of the Densenet architecture you'll use in this week's graded assignment!

In [ ]:
```
```