# AI for Medicine Course 1 Week 1 lecture exercises
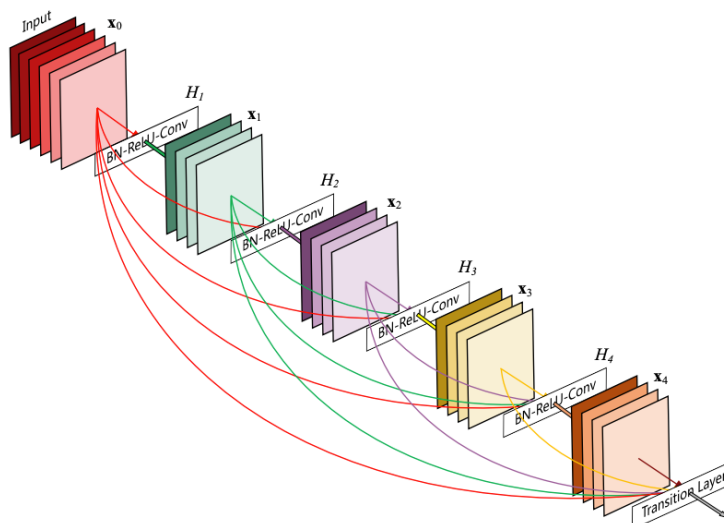
# Densenet

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 [3]:
```python
# Create the base pre-trained model
base_model = DenseNet121(weights='./nih/densenet.hdf5', include_top=False);
```

View a summary of the model

In [4]:
```python
# Print the model summary
base_model.summary()
```

```
Model: "densenet121"
_____
_____
Layer (type)                  Output Shape         Param #     Connected t
o
================================================================
=====================
input_2 (InputLayer)          (None, None, None, 3 0
_____
_____
zero_padding2d_3 (ZeroPadding2D (None, None, None, 3 0           input_2[0]
[0]
_____
_____
conv1/conv (Conv2D)           (None, None, None, 6 9408        zero_paddin
g2d_3[0][0]
_____
_____
conv1/bn (BatchNormalization)  (None, None, None, 6 256         conv1/conv
[0][0]
```

In [5]:
```python
# Print out the first five layers
layers_l = base_model.layers

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

```
First 5 layers
```

Out[5]:
```
[<keras.engine.input_layer.InputLayer at 0x7f926c0d91d0>,
 <keras.layers.convolutional.ZeroPadding2D at 0x7f926c0f2c18>,
 <keras.layers.convolutional.Conv2D at 0x7f926c0f8978>,
 <keras.layers.normalization.BatchNormalization at 0x7f9244334198>,
 <keras.layers.core.Activation at 0x7f9244334208>]
```

In [7]:
```python
# Print out the last five layers
print("Last 5 layers")
layers_l[-6:-1]
```

```
Last 5 layers
```

Out[7]:
```
[<keras.layers.normalization.BatchNormalization at 0x7f91c47f1a90>,
 <keras.layers.core.Activation at 0x7f91c47f1da0>,
 <keras.layers.convolutional.Conv2D at 0x7f91c4780e48>,
 <keras.layers.merge.Concatenate at 0x7f91c47aa8d0>,
 <keras.layers.normalization.BatchNormalization at 0x7f91c47aa9e8>]
```

In [8]:
```python
# 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[8]:
```
[<keras.layers.convolutional.Conv2D at 0x7f926c0f8978>,
 <keras.layers.convolutional.Conv2D at 0x7f92443ea048>,
 <keras.layers.convolutional.Conv2D at 0x7f92444ad3c8>,
 <keras.layers.convolutional.Conv2D at 0x7f924424beb8>,
 <keras.layers.convolutional.Conv2D at 0x7f924426fcc0>]
```

In [9]:
```python
# Print out the total number of convolutional layers
print(f"There are {len(conv2D_layers)} convolutional layers")
```

There are 120 convolutional layers

In [10]:
```python
# Print the number of channels in the input
print("The input has 3 channels")
base_model.input
```

The input has 3 channels

Out[10]: `<tf.Tensor 'input_2:0' shape=(?, ?, ?, 3) dtype=float32>`

In [11]:
```python
# Print the number of output channels
print("The output has 1024 channels")
x = base_model.output
x
```

The output has 1024 channels

Out[11]: `<tf.Tensor 'relu_1/Relu:0' shape=(?, ?, ?, 1024) dtype=float32>`

In [12]:
```python
# Add a global spatial average pooling layer
x_pool = GlobalAveragePooling2D()(x)
x_pool
```

Out[12]: `<tf.Tensor 'global_average_pooling2d_1/Mean:0' shape=(?, 1024) dtype=float32>`

In [13]:
```python
# 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} classes")
```

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

In [15]:
```python
# Add a logistic layer the same size as the number of classes you're trying to pre
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[15]: `<tf.Tensor 'dense_2/Sigmoid:0' shape=(?, 5) dtype=float32>`

In [19]:
```python
# Create an updated model
model = Model(inputs=base_model.input, outputs=predictions)
```

In [20]:
```python
# 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 [ ]: