HW5_Task2

April 30, 2019

1 Applied Machine Learning Homework 5

1.0.1 Po-Chieh Liu (pl2441), Peter Grantcharov (pdg2116)

2 Task 2

Train a **multilayer perceptron (fully connected)** on the **Fashion MNIST dataset** using the **traditional train/test** split as given by **fashion_mnist.load_data in keras**. Use a separate **10000 samples (from the training set) for model selection** and to compute **learning curves** (**accuracy vs epochs**, not vs n_samples). Compare a "vanilla" model with a model using **drop-out** (potentially a bigger model), and to a model using **batch normalization** and **residual connections** (but not dropout). Visualize learning curves for all models.

```
In [0]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
```

```
In [2]: from keras.datasets import fashion_mnist
        (X_train, y_train), (X_test, y_test) = fashion_mnist.load_data()
```

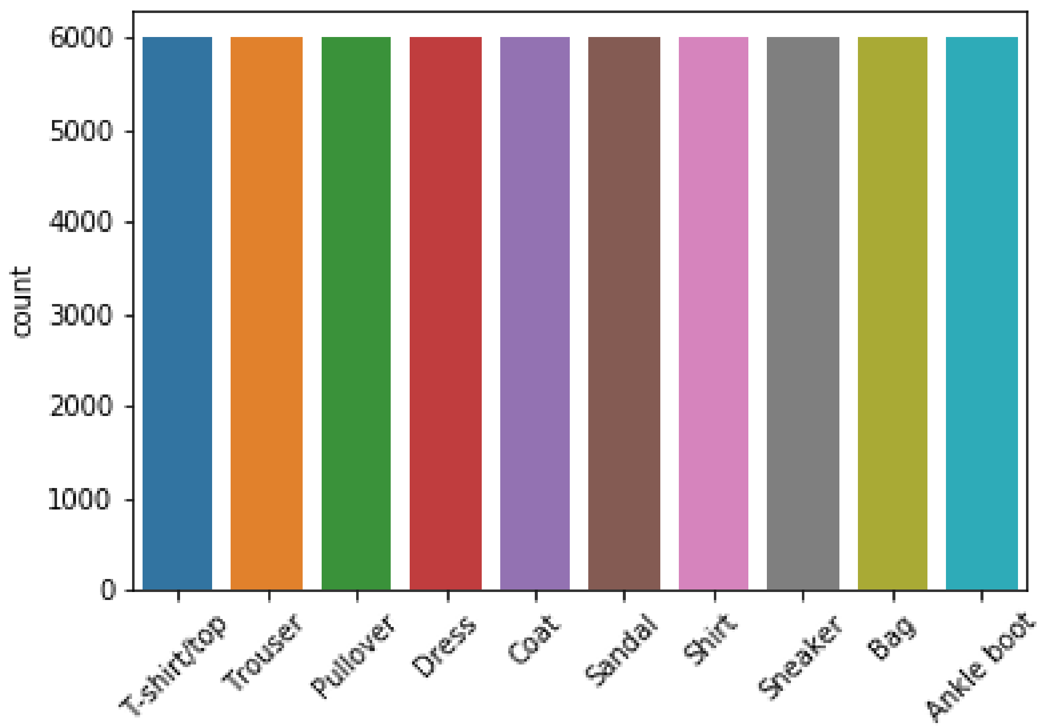
```
Using TensorFlow backend.


Downloading data from http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-labels-i
32768/29515 [==================================] - 0s 3us/step
Downloading data from http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-images-i
26427392/26421880 [==============================] - 2s 0us/step
Downloading data from http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-labels-id
8192/5148 [===============================================] - 0s 0us/step
Downloading data from http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-images-id
4423680/4422102 [==============================] - 1s 0us/step
```

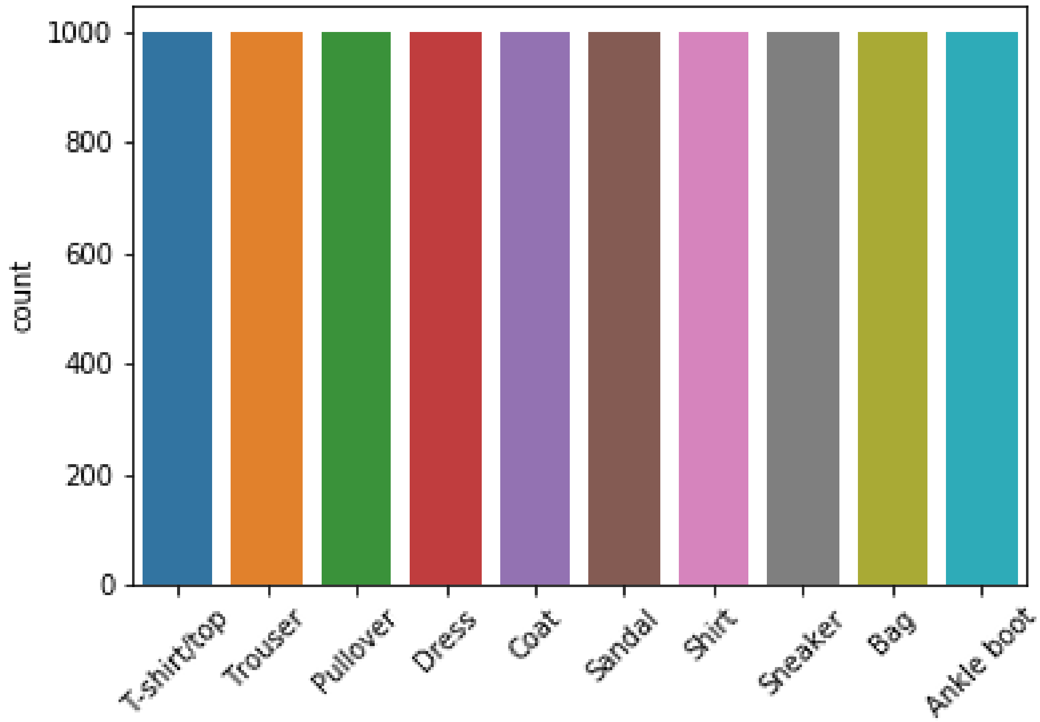
```
In [0]: # scale
        X_train = X_train/255
        X_test = X_test/255
```

Confirm that we have a balanced data set:

In [4]:
```
ax = sns.countplot(y_train)
_= ax.set_xticklabels(['T-shirt/top','Trouser','Pullover',
                       'Dress','Coat','Sandal','Shirt',
                       'Sneaker','Bag','Ankle boot'],
                      rotation=45)
```



In [5]:
```
ax = sns.countplot(y_test)
_= ax.set_xticklabels(['T-shirt/top','Trouser','Pullover',
                       'Dress','Coat','Sandal','Shirt',
                       'Sneaker','Bag','Ankle boot'],
                      rotation=45)
```



Check GPU status

In [6]:
```
# tf
import tensorflow as tf
sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))
from keras import backend
backend.tensorflow_backend._get_available_gpus()
```

Out[6]:
```
['/job:localhost/replica:0/task:0/device:GPU:0']
```

3 Base 'Vanilla' Model

In [0]:
```
from keras import Sequential
from keras.layers import Dense, Flatten, Dropout, BatchNormalization, Input, add, Acti
from keras import regularizers
from keras import Model
```

In [8]:
```
# 1 layer model
# initiate model
model_vanilla1 = Sequential()

# flatten layer
model_vanilla1.add(Flatten(input_shape = (28,28)))
```

```python
# first layer
model_vanilla1.add(Dense(128,
                input_dim = 784,
                activation='relu'))

# output layer
model_vanilla1.add(Dense(10, activation='softmax'))

# compile
model_vanilla1.compile(optimizer = 'adam',
              loss = 'sparse_categorical_crossentropy',
              metrics = ['accuracy'])
model_vanilla1.summary()
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/framework/op_
Instructions for updating:
Colocations handled automatically by placer.
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_1 (Flatten)          (None, 784)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               100480
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290
=================================================================
Total params: 101,770
Trainable params: 101,770
Non-trainable params: 0
_________________________________________________________________
```

In [10]:
```python
vanilla1 = model_vanilla1.fit(X_train,
                              y_train,
                              epochs=50,
                              validation_split=10000/60000)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/math_ops.
Instructions for updating:
Use tf.cast instead.
Train on 50000 samples, validate on 10000 samples
Epoch 1/50
50000/50000 [==============================] - 5s 94us/step - loss: 0.5173 - acc: 0.8194 - val_
Epoch 2/50
50000/50000 [==============================] - 4s 89us/step - loss: 0.3885 - acc: 0.8603 - val_
Epoch 3/50
50000/50000 [==============================] - 4s 89us/step - loss: 0.3461 - acc: 0.8744 - val_
Epoch 4/50
```

```
50000/50000 [==============================] - 4s 88us/step - loss: 0.3189 - acc: 0.8836 - val_
Epoch 5/50
50000/50000 [==============================] - 4s 88us/step - loss: 0.3021 - acc: 0.8898 - val_
Epoch 6/50
50000/50000 [==============================] - 4s 88us/step - loss: 0.2831 - acc: 0.8954 - val_
Epoch 7/50
50000/50000 [==============================] - 4s 88us/step - loss: 0.2728 - acc: 0.8985 - val_
Epoch 8/50
50000/50000 [==============================] - 4s 89us/step - loss: 0.2594 - acc: 0.9042 - val_
Epoch 9/50
50000/50000 [==============================] - 4s 89us/step - loss: 0.2519 - acc: 0.9064 - val_
Epoch 10/50
50000/50000 [==============================] - 4s 89us/step - loss: 0.2422 - acc: 0.9087 - val_
Epoch 11/50
50000/50000 [==============================] - 4s 88us/step - loss: 0.2329 - acc: 0.9141 - val_
Epoch 12/50
50000/50000 [==============================] - 4s 88us/step - loss: 0.2244 - acc: 0.9157 - val_
Epoch 13/50
50000/50000 [==============================] - 4s 88us/step - loss: 0.2191 - acc: 0.9176 - val_
Epoch 14/50
50000/50000 [==============================] - 4s 88us/step - loss: 0.2122 - acc: 0.9198 - val_
Epoch 15/50
50000/50000 [==============================] - 5s 99us/step - loss: 0.2065 - acc: 0.9239 - val_
Epoch 16/50
50000/50000 [==============================] - 5s 99us/step - loss: 0.1991 - acc: 0.9257 - val_
Epoch 17/50
50000/50000 [==============================] - 4s 88us/step - loss: 0.1928 - acc: 0.9280 - val_
Epoch 18/50
50000/50000 [==============================] - 4s 86us/step - loss: 0.1894 - acc: 0.9288 - val_
Epoch 19/50
50000/50000 [==============================] - 4s 87us/step - loss: 0.1819 - acc: 0.9323 - val_
Epoch 20/50
50000/50000 [==============================] - 5s 97us/step - loss: 0.1776 - acc: 0.9336 - val_
Epoch 21/50
50000/50000 [==============================] - 5s 93us/step - loss: 0.1747 - acc: 0.9351 - val_
Epoch 22/50
50000/50000 [==============================] - 4s 89us/step - loss: 0.1703 - acc: 0.9368 - val_
Epoch 23/50
50000/50000 [==============================] - 4s 88us/step - loss: 0.1655 - acc: 0.9375 - val_
Epoch 24/50
50000/50000 [==============================] - 4s 88us/step - loss: 0.1605 - acc: 0.9393 - val_
Epoch 25/50
50000/50000 [==============================] - 4s 88us/step - loss: 0.1583 - acc: 0.9410 - val_
Epoch 26/50
50000/50000 [==============================] - 4s 88us/step - loss: 0.1538 - acc: 0.9426 - val_
Epoch 27/50
50000/50000 [==============================] - 4s 87us/step - loss: 0.1491 - acc: 0.9442 - val_
Epoch 28/50
```

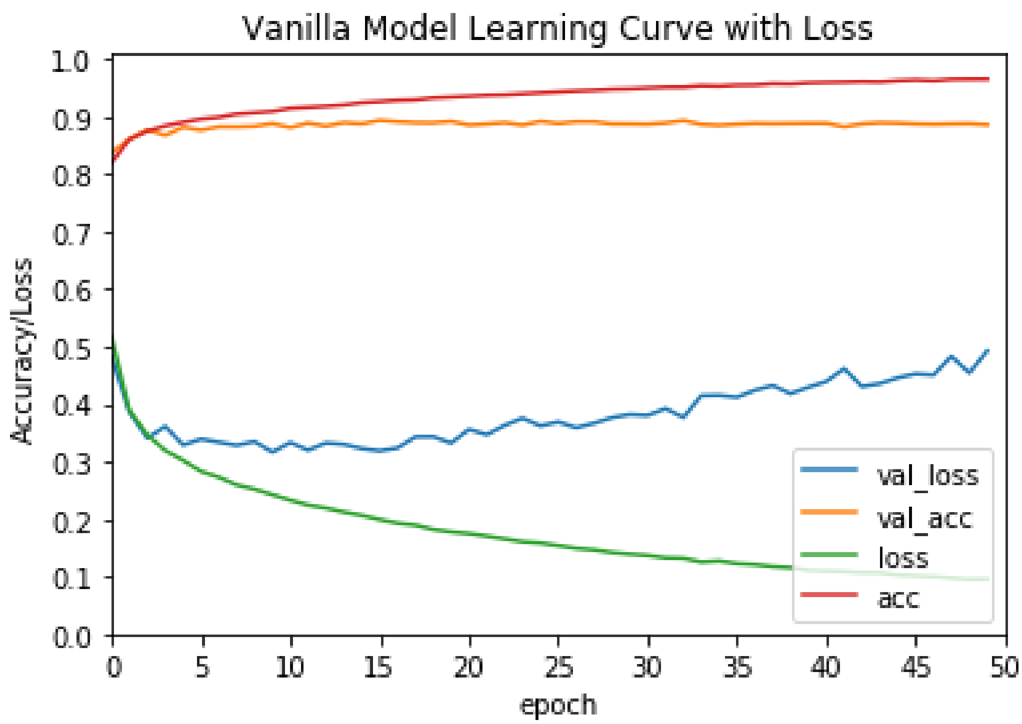
```
50000/50000 [==============================] - 4s 88us/step - loss: 0.1465 - acc: 0.9453 - val_
Epoch 29/50
50000/50000 [==============================] - 4s 88us/step - loss: 0.1419 - acc: 0.9471 - val_
Epoch 30/50
50000/50000 [==============================] - 4s 87us/step - loss: 0.1390 - acc: 0.9477 - val_
Epoch 31/50
50000/50000 [==============================] - 4s 87us/step - loss: 0.1368 - acc: 0.9492 - val_
Epoch 32/50
50000/50000 [==============================] - 4s 87us/step - loss: 0.1324 - acc: 0.9505 - val_
Epoch 33/50
50000/50000 [==============================] - 5s 99us/step - loss: 0.1318 - acc: 0.9507 - val_
Epoch 34/50
50000/50000 [==============================] - 5s 98us/step - loss: 0.1252 - acc: 0.9531 - val_
Epoch 35/50
50000/50000 [==============================] - 4s 88us/step - loss: 0.1271 - acc: 0.9524 - val_
Epoch 36/50
50000/50000 [==============================] - 4s 89us/step - loss: 0.1226 - acc: 0.9542 - val_
Epoch 37/50
50000/50000 [==============================] - 4s 87us/step - loss: 0.1209 - acc: 0.9545 - val_
Epoch 38/50
50000/50000 [==============================] - 4s 88us/step - loss: 0.1173 - acc: 0.9567 - val_
Epoch 39/50
50000/50000 [==============================] - 4s 89us/step - loss: 0.1155 - acc: 0.9561 - val_
Epoch 40/50
50000/50000 [==============================] - 4s 88us/step - loss: 0.1109 - acc: 0.9582 - val_
Epoch 41/50
50000/50000 [==============================] - 4s 87us/step - loss: 0.1097 - acc: 0.9589 - val_
Epoch 42/50
50000/50000 [==============================] - 4s 87us/step - loss: 0.1087 - acc: 0.9594 - val_
Epoch 43/50
50000/50000 [==============================] - 4s 88us/step - loss: 0.1065 - acc: 0.9602 - val_
Epoch 44/50
50000/50000 [==============================] - 4s 87us/step - loss: 0.1066 - acc: 0.9597 - val_
Epoch 45/50
50000/50000 [==============================] - 4s 88us/step - loss: 0.1026 - acc: 0.9618 - val_
Epoch 46/50
50000/50000 [==============================] - 4s 88us/step - loss: 0.1010 - acc: 0.9630 - val_
Epoch 47/50
50000/50000 [==============================] - 4s 88us/step - loss: 0.0998 - acc: 0.9618 - val_
Epoch 48/50
50000/50000 [==============================] - 4s 89us/step - loss: 0.0974 - acc: 0.9641 - val_
Epoch 49/50
50000/50000 [==============================] - 4s 87us/step - loss: 0.0951 - acc: 0.9645 - val_
Epoch 50/50
50000/50000 [==============================] - 4s 89us/step - loss: 0.0957 - acc: 0.9646 - val_
```

```
In [30]: print('Vanilla modeltest score:',
```

```
               model_vanilla1.evaluate(X_test, y_test))
```

```
10000/10000 [==============================] - 0s 40us/step
Vanilla modeltest score: [0.5406438440233469, 0.8806]
```

```
In [27]: _ = pd.DataFrame(vanilla1.history).plot(
             title = 'Vanilla Model Learning Curve with Loss',
             xticks = range(0,51,5),
             yticks = [0.1* x for x in range(0,11)]
         )
         _ = plt.legend(loc = 4)
         _ = plt.xlabel('epoch')
         _ = plt.ylabel('Accuracy/Loss')
```



Quick observation Validation accuracy starts to flatten at around 5 epochs, while the validation loss bottoms out at about epoch 5, and thereafter steadily increases until the end.

Larger and Deeper Vanilla Model 6 layers with 512 cells

```
In [28]: # 6 layers model with 512 cells
         # initiate model
         model_vanilla2 = Sequential()
```

```python
# flatten layer
model_vanilla2.add(Flatten(input_shape = (28,28)))

# first layer
model_vanilla2.add(Dense(512,
                input_dim = 784,
                activation='relu'))

# second
model_vanilla2.add(Dense(512,
                activation='relu'))

# third
model_vanilla2.add(Dense(512,
                activation='relu'))

# fourth
model_vanilla2.add(Dense(512,
                activation='relu'))

# fifth
model_vanilla2.add(Dense(512,
                activation='relu'))

# sixth
model_vanilla2.add(Dense(512,
                activation='relu'))

# output layer
model_vanilla2.add(Dense(10, activation='softmax'))

# compile
model_vanilla2.compile(optimizer = 'adam',
              loss = 'sparse_categorical_crossentropy',
              metrics = ['accuracy'])
model_vanilla2.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_2 (Flatten)          (None, 784)               0
_________________________________________________________________
dense_3 (Dense)              (None, 512)               401920
_________________________________________________________________
dense_4 (Dense)              (None, 512)               262656
_________________________________________________________________
dense_5 (Dense)              (None, 512)               262656
```

```
_________________________________________________________________
dense_6 (Dense)              (None, 512)               262656
_________________________________________________________________
dense_7 (Dense)              (None, 512)               262656
_________________________________________________________________
dense_8 (Dense)              (None, 512)               262656
_________________________________________________________________
dense_9 (Dense)              (None, 10)                5130
=================================================================
Total params: 1,720,330
Trainable params: 1,720,330
Non-trainable params: 0
_________________________________________________________________
```

```
In [29]: vanilla2 = model_vanilla2.fit(X_train,
                                       y_train,
                                       epochs=50,
                                       validation_split= 10000/60000)
```

```
Train on 50000 samples, validate on 10000 samples
Epoch 1/50
50000/50000 [==============================] - 8s 153us/step - loss: 0.5558 - acc: 0.7994 - val
Epoch 2/50
50000/50000 [==============================] - 7s 142us/step - loss: 0.4151 - acc: 0.8527 - val
Epoch 3/50
50000/50000 [==============================] - 7s 143us/step - loss: 0.3781 - acc: 0.8652 - val
Epoch 4/50
50000/50000 [==============================] - 7s 143us/step - loss: 0.3495 - acc: 0.8744 - val
Epoch 5/50
50000/50000 [==============================] - 8s 155us/step - loss: 0.3254 - acc: 0.8822 - val
Epoch 6/50
50000/50000 [==============================] - 8s 151us/step - loss: 0.3159 - acc: 0.8876 - val
Epoch 7/50
50000/50000 [==============================] - 8s 157us/step - loss: 0.2969 - acc: 0.8938 - val
Epoch 8/50
50000/50000 [==============================] - 7s 142us/step - loss: 0.2927 - acc: 0.8929 - val
Epoch 9/50
50000/50000 [==============================] - 7s 143us/step - loss: 0.2780 - acc: 0.8997 - val
Epoch 10/50
50000/50000 [==============================] - 7s 142us/step - loss: 0.2698 - acc: 0.9004 - val
Epoch 11/50
50000/50000 [==============================] - 7s 142us/step - loss: 0.2635 - acc: 0.9035 - val
Epoch 12/50
50000/50000 [==============================] - 7s 142us/step - loss: 0.2587 - acc: 0.9074 - val
Epoch 13/50
50000/50000 [==============================] - 7s 142us/step - loss: 0.2476 - acc: 0.9104 - val
Epoch 14/50
```

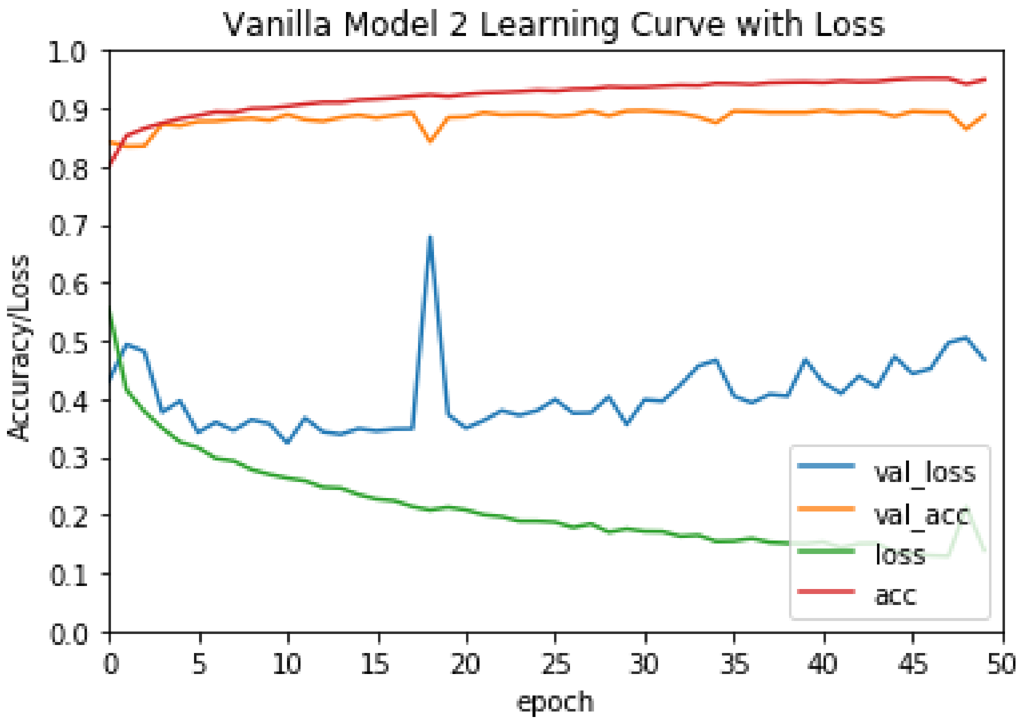
```
50000/50000 [==============================] - 7s 142us/step - loss: 0.2461 - acc: 0.9104 - val
Epoch 15/50
50000/50000 [==============================] - 7s 142us/step - loss: 0.2349 - acc: 0.9144 - val
Epoch 16/50
50000/50000 [==============================] - 7s 142us/step - loss: 0.2272 - acc: 0.9160 - val
Epoch 17/50
50000/50000 [==============================] - 7s 149us/step - loss: 0.2243 - acc: 0.9180 - val
Epoch 18/50
50000/50000 [==============================] - 8s 154us/step - loss: 0.2142 - acc: 0.9209 - val
Epoch 19/50
50000/50000 [==============================] - 7s 142us/step - loss: 0.2083 - acc: 0.9232 - val
Epoch 20/50
50000/50000 [==============================] - 7s 142us/step - loss: 0.2131 - acc: 0.9206 - val
Epoch 21/50
50000/50000 [==============================] - 7s 141us/step - loss: 0.2084 - acc: 0.9241 - val
Epoch 22/50
50000/50000 [==============================] - 7s 141us/step - loss: 0.2001 - acc: 0.9265 - val
Epoch 23/50
50000/50000 [==============================] - 7s 141us/step - loss: 0.1968 - acc: 0.9275 - val
Epoch 24/50
50000/50000 [==============================] - 7s 141us/step - loss: 0.1892 - acc: 0.9290 - val
Epoch 25/50
50000/50000 [==============================] - 7s 142us/step - loss: 0.1892 - acc: 0.9311 - val
Epoch 26/50
50000/50000 [==============================] - 7s 141us/step - loss: 0.1875 - acc: 0.9300 - val
Epoch 27/50
50000/50000 [==============================] - 7s 141us/step - loss: 0.1783 - acc: 0.9330 - val
Epoch 28/50
50000/50000 [==============================] - 8s 157us/step - loss: 0.1845 - acc: 0.9336 - val
Epoch 29/50
50000/50000 [==============================] - 8s 161us/step - loss: 0.1702 - acc: 0.9372 - val
Epoch 30/50
50000/50000 [==============================] - 7s 142us/step - loss: 0.1759 - acc: 0.9362 - val
Epoch 31/50
50000/50000 [==============================] - 7s 141us/step - loss: 0.1716 - acc: 0.9366 - val
Epoch 32/50
50000/50000 [==============================] - 7s 142us/step - loss: 0.1712 - acc: 0.9383 - val
Epoch 33/50
50000/50000 [==============================] - 7s 141us/step - loss: 0.1631 - acc: 0.9398 - val
Epoch 34/50
50000/50000 [==============================] - 7s 141us/step - loss: 0.1648 - acc: 0.9390 - val
Epoch 35/50
50000/50000 [==============================] - 7s 140us/step - loss: 0.1543 - acc: 0.9431 - val
Epoch 36/50
50000/50000 [==============================] - 7s 140us/step - loss: 0.1549 - acc: 0.9425 - val
Epoch 37/50
50000/50000 [==============================] - 7s 141us/step - loss: 0.1593 - acc: 0.9412 - val
Epoch 38/50
```

```
50000/50000 [==============================] - 7s 142us/step - loss: 0.1525 - acc: 0.9441 - val
Epoch 39/50
50000/50000 [==============================] - 7s 142us/step - loss: 0.1509 - acc: 0.9449 - val
Epoch 40/50
50000/50000 [==============================] - 8s 156us/step - loss: 0.1502 - acc: 0.9454 - val
Epoch 41/50
50000/50000 [==============================] - 7s 144us/step - loss: 0.1530 - acc: 0.9444 - val
Epoch 42/50
50000/50000 [==============================] - 7s 141us/step - loss: 0.1437 - acc: 0.9469 - val
Epoch 43/50
50000/50000 [==============================] - 7s 140us/step - loss: 0.1504 - acc: 0.9456 - val
Epoch 44/50
50000/50000 [==============================] - 7s 140us/step - loss: 0.1508 - acc: 0.9461 - val
Epoch 45/50
50000/50000 [==============================] - 7s 141us/step - loss: 0.1357 - acc: 0.9496 - val
Epoch 46/50
50000/50000 [==============================] - 7s 141us/step - loss: 0.1331 - acc: 0.9515 - val
Epoch 47/50
50000/50000 [==============================] - 7s 149us/step - loss: 0.1299 - acc: 0.9518 - val
Epoch 48/50
50000/50000 [==============================] - 7s 148us/step - loss: 0.1288 - acc: 0.9517 - val
Epoch 49/50
50000/50000 [==============================] - 7s 141us/step - loss: 0.2142 - acc: 0.9419 - val
Epoch 50/50
50000/50000 [==============================] - 7s 141us/step - loss: 0.1390 - acc: 0.9492 - val
```

```
In [31]: print('Vanilla model 2 test score:',
               model_vanilla2.evaluate(X_test, y_test))
```

```
10000/10000 [==============================] - 0s 46us/step
Vanilla model 2 test score: [0.49062124730870127, 0.8858]
```

```
In [32]: _ = pd.DataFrame(vanilla2.history).plot(
             title = 'Vanilla Model 2 Learning Curve with Loss',
             xticks = range(0,51,5),
             yticks = [0.1* x for x in range(0,11)]
         )
         _ = plt.legend(loc = 4)
         _ = plt.xlabel('epoch')
         _ = plt.ylabel('Accuracy/Loss')
```



Quick observation For the deeper and larger vanilla model, the validation accuracy also starts to flatten at around the fifth epoch, while the validation loss seems to bottom out at around 15. It then has a very odd spike, but seems to correct itself and then steadily rise after that.

4 Dropout Model

In [33]:
```
# 1 layer
# initiate model
model_dropout1 = Sequential()

# flatten layer
model_dropout1.add(Flatten(input_shape = (28,28)))

# first layer
model_dropout1.add(Dense(128,
                input_dim = 784,
                activation='relu'))

# drop out layer
model_dropout1.add(Dropout(rate = 0.5))

# output layer
```

```
model_dropout1.add(Dense(10, activation='softmax'))

# compile
model_dropout1.compile(optimizer = 'adam',
                loss = 'sparse_categorical_crossentropy',
                metrics = ['accuracy'])

model_dropout1.summary()
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_3 (Flatten)          (None, 784)               0
_________________________________________________________________
dense_10 (Dense)             (None, 128)               100480
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_11 (Dense)             (None, 10)                1290
=================================================================
Total params: 101,770
Trainable params: 101,770
Non-trainable params: 0
_________________________________________________________________
```

```
In [34]: dropout1 = model_dropout1.fit(X_train,
                               y_train,
                               epochs=50,
                               validation_split= 10000/60000)
```

```
Train on 50000 samples, validate on 10000 samples
Epoch 1/50
50000/50000 [==============================] - 5s 96us/step - loss: 0.6433 - acc: 0.7741 - val_
Epoch 2/50
50000/50000 [==============================] - 5s 92us/step - loss: 0.4833 - acc: 0.8257 - val_
Epoch 3/50
50000/50000 [==============================] - 5s 92us/step - loss: 0.4494 - acc: 0.8379 - val_
Epoch 4/50
50000/50000 [==============================] - 5s 92us/step - loss: 0.4260 - acc: 0.8448 - val_
Epoch 5/50
50000/50000 [==============================] - 5s 92us/step - loss: 0.4111 - acc: 0.8506 - val_
Epoch 6/50
50000/50000 [==============================] - 5s 92us/step - loss: 0.4002 - acc: 0.8541 - val_
Epoch 7/50
```

```
50000/50000 [==============================] - 5s 92us/step - loss: 0.3900 - acc: 0.8578 - val_
Epoch 8/50
50000/50000 [==============================] - 5s 92us/step - loss: 0.3797 - acc: 0.8610 - val_
Epoch 9/50
50000/50000 [==============================] - 5s 91us/step - loss: 0.3762 - acc: 0.8624 - val_
Epoch 10/50
50000/50000 [==============================] - 5s 92us/step - loss: 0.3712 - acc: 0.8644 - val_
Epoch 11/50
50000/50000 [==============================] - 5s 92us/step - loss: 0.3625 - acc: 0.8681 - val_
Epoch 12/50
50000/50000 [==============================] - 5s 98us/step - loss: 0.3551 - acc: 0.8687 - val_
Epoch 13/50
50000/50000 [==============================] - 6s 111us/step - loss: 0.3545 - acc: 0.8687 - val
Epoch 14/50
50000/50000 [==============================] - 5s 100us/step - loss: 0.3512 - acc: 0.8696 - val
Epoch 15/50
50000/50000 [==============================] - 5s 99us/step - loss: 0.3489 - acc: 0.8724 - val_
Epoch 16/50
50000/50000 [==============================] - 5s 105us/step - loss: 0.3407 - acc: 0.8728 - val
Epoch 17/50
50000/50000 [==============================] - 5s 96us/step - loss: 0.3387 - acc: 0.8743 - val_
Epoch 18/50
50000/50000 [==============================] - 5s 91us/step - loss: 0.3349 - acc: 0.8759 - val_
Epoch 19/50
50000/50000 [==============================] - 5s 92us/step - loss: 0.3352 - acc: 0.8756 - val_
Epoch 20/50
50000/50000 [==============================] - 5s 94us/step - loss: 0.3307 - acc: 0.8764 - val_
Epoch 21/50
50000/50000 [==============================] - 5s 92us/step - loss: 0.3260 - acc: 0.8797 - val_
Epoch 22/50
50000/50000 [==============================] - 5s 93us/step - loss: 0.3241 - acc: 0.8788 - val_
Epoch 23/50
50000/50000 [==============================] - 5s 92us/step - loss: 0.3219 - acc: 0.8795 - val_
Epoch 24/50
50000/50000 [==============================] - 5s 92us/step - loss: 0.3179 - acc: 0.8827 - val_
Epoch 25/50
50000/50000 [==============================] - 5s 93us/step - loss: 0.3163 - acc: 0.8798 - val_
Epoch 26/50
50000/50000 [==============================] - 5s 94us/step - loss: 0.3129 - acc: 0.8836 - val_
Epoch 27/50
50000/50000 [==============================] - 5s 93us/step - loss: 0.3145 - acc: 0.8816 - val_
Epoch 28/50
50000/50000 [==============================] - 5s 93us/step - loss: 0.3113 - acc: 0.8844 - val_
Epoch 29/50
50000/50000 [==============================] - 5s 93us/step - loss: 0.3102 - acc: 0.8825 - val_
Epoch 30/50
50000/50000 [==============================] - 5s 93us/step - loss: 0.3109 - acc: 0.8839 - val_
Epoch 31/50
```

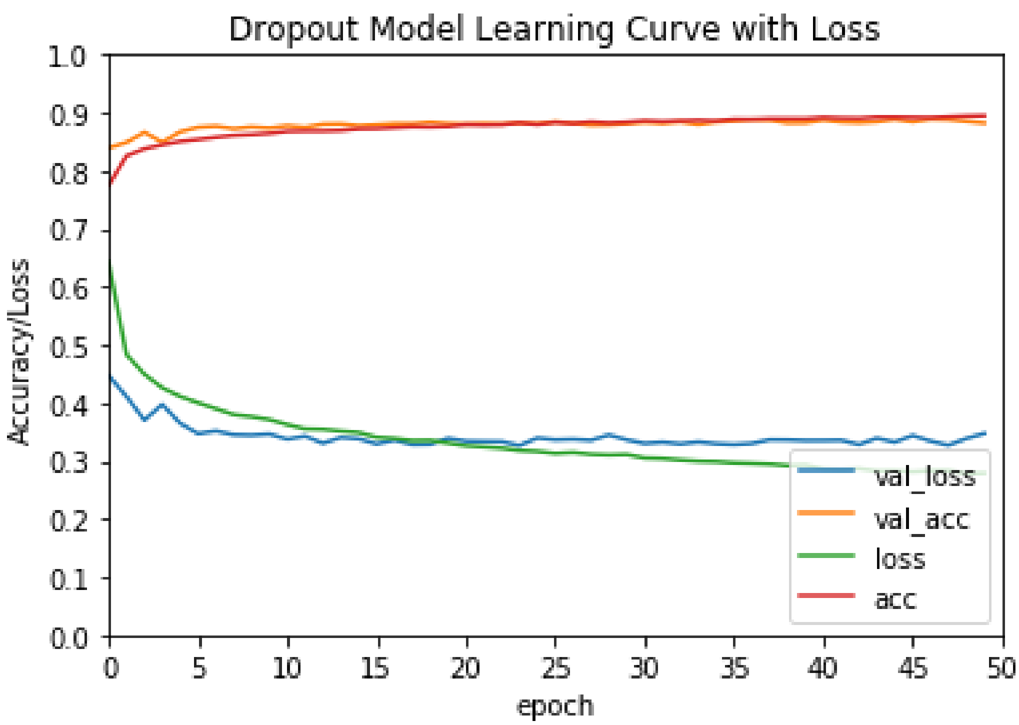
```
50000/50000 [==============================] - 5s 94us/step - loss: 0.3050 - acc: 0.8863 - val_
Epoch 32/50
50000/50000 [==============================] - 5s 99us/step - loss: 0.3041 - acc: 0.8851 - val_
Epoch 33/50
50000/50000 [==============================] - 5s 105us/step - loss: 0.3016 - acc: 0.8862 - va
Epoch 34/50
50000/50000 [==============================] - 5s 97us/step - loss: 0.2996 - acc: 0.8872 - val_
Epoch 35/50
50000/50000 [==============================] - 5s 92us/step - loss: 0.2989 - acc: 0.8860 - val_
Epoch 36/50
50000/50000 [==============================] - 5s 91us/step - loss: 0.2967 - acc: 0.8889 - val_
Epoch 37/50
50000/50000 [==============================] - 5s 91us/step - loss: 0.2959 - acc: 0.8886 - val_
Epoch 38/50
50000/50000 [==============================] - 5s 91us/step - loss: 0.2947 - acc: 0.8898 - val_
Epoch 39/50
50000/50000 [==============================] - 5s 91us/step - loss: 0.2921 - acc: 0.8898 - val_
Epoch 40/50
50000/50000 [==============================] - 5s 95us/step - loss: 0.2931 - acc: 0.8897 - val_
Epoch 41/50
50000/50000 [==============================] - 5s 105us/step - loss: 0.2866 - acc: 0.8918 - va
Epoch 42/50
50000/50000 [==============================] - 5s 91us/step - loss: 0.2854 - acc: 0.8908 - val_
Epoch 43/50
50000/50000 [==============================] - 5s 91us/step - loss: 0.2869 - acc: 0.8904 - val_
Epoch 44/50
50000/50000 [==============================] - 5s 92us/step - loss: 0.2829 - acc: 0.8923 - val_
Epoch 45/50
50000/50000 [==============================] - 5s 91us/step - loss: 0.2830 - acc: 0.8923 - val_
Epoch 46/50
50000/50000 [==============================] - 5s 93us/step - loss: 0.2816 - acc: 0.8918 - val_
Epoch 47/50
50000/50000 [==============================] - 5s 92us/step - loss: 0.2833 - acc: 0.8914 - val_
Epoch 48/50
50000/50000 [==============================] - 5s 92us/step - loss: 0.2836 - acc: 0.8930 - val_
Epoch 49/50
50000/50000 [==============================] - 5s 95us/step - loss: 0.2783 - acc: 0.8942 - val_
Epoch 50/50
50000/50000 [==============================] - 5s 104us/step - loss: 0.2791 - acc: 0.8949 - va
```

```
In [35]: print('Dropout model 1 test score:',
               model_dropout1.evaluate(X_test, y_test))
```

```
10000/10000 [==============================] - 0s 44us/step
Dropout model 1 test score: [0.36197602721452715, 0.883]
```

```
In [37]: _ = pd.DataFrame(dropout1.history).plot(
```

```
    title = 'Dropout Model Learning Curve with Loss',
    xticks = range(0,51,5),
    yticks = [0.1* x for x in range(0,11)]
)
_ = plt.legend(loc = 4)
_ = plt.xlabel('epoch')
_ = plt.ylabel('Accuracy/Loss')
```



Quick observation Both training and validation accuracy flatten at a level around 0.85~0.9. In comparison with the vanilla model, the training accuracy doesn't appear to be overfitted which reached 1, and also saw the validation loss steadily rise as it was approaching 1.

Larger and deeper dropout model

```
In [38]: # initiate model
         model_dropout2 = Sequential()

         # flatten layer
         model_dropout2.add(Flatten(input_shape = (28,28)))

         # first layer
         model_dropout2.add(Dense(512,
                         input_dim = 784,
```

```
                activation='relu'))

# drop out layer
model_dropout2.add(Dropout(rate = 0.5))

# second layer
model_dropout2.add(Dense(512,
                activation='relu'))

# drop out layer
model_dropout2.add(Dropout(rate = 0.5))

# third layer
model_dropout2.add(Dense(512,
                activation='relu'))

# drop out layer
model_dropout2.add(Dropout(rate = 0.5))

# fourth layer
model_dropout2.add(Dense(512,
                activation='relu'))

# drop out layer
model_dropout2.add(Dropout(rate = 0.5))

# fifth layer
model_dropout2.add(Dense(512,
                activation='relu'))

# drop out layer
model_dropout2.add(Dropout(rate = 0.5))

# sixth layer
model_dropout2.add(Dense(512,
                activation='relu'))

# drop out layer
model_dropout2.add(Dropout(rate = 0.5))

# output layer
model_dropout2.add(Dense(10, activation='softmax'))

# compile
model_dropout2.compile(optimizer = 'adam',
              loss = 'sparse_categorical_crossentropy',
              metrics = ['accuracy'])
```

```
model_dropout2.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_4 (Flatten)          (None, 784)               0
_________________________________________________________________
dense_12 (Dense)             (None, 512)               401920
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_13 (Dense)             (None, 512)               262656
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_14 (Dense)             (None, 512)               262656
_________________________________________________________________
dropout_4 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_15 (Dense)             (None, 512)               262656
_________________________________________________________________
dropout_5 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_16 (Dense)             (None, 512)               262656
_________________________________________________________________
dropout_6 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_17 (Dense)             (None, 512)               262656
_________________________________________________________________
dropout_7 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_18 (Dense)             (None, 10)                5130
=================================================================
Total params: 1,720,330
Trainable params: 1,720,330
Non-trainable params: 0
_________________________________________________________________
```

```
In [39]: dropout2 = model_dropout2.fit(X_train,
                                       y_train,
                                       epochs=50,
                                       validation_split=10000/60000)
```

```
Train on 50000 samples, validate on 10000 samples
Epoch 1/50
50000/50000 [==============================] - 9s 189us/step - loss: 1.0096 - acc: 0.6020 - va
Epoch 2/50
```

```
50000/50000 [==============================] - 8s 166us/step - loss: 0.7129 - acc: 0.7362 - va
Epoch 3/50
50000/50000 [==============================] - 8s 158us/step - loss: 0.6647 - acc: 0.7648 - va
Epoch 4/50
50000/50000 [==============================] - 8s 157us/step - loss: 0.6460 - acc: 0.7766 - va
Epoch 5/50
50000/50000 [==============================] - 8s 157us/step - loss: 0.6257 - acc: 0.7862 - va
Epoch 6/50
50000/50000 [==============================] - 8s 158us/step - loss: 0.6193 - acc: 0.7891 - va
Epoch 7/50
50000/50000 [==============================] - 8s 157us/step - loss: 0.6091 - acc: 0.7930 - va
Epoch 8/50
50000/50000 [==============================] - 8s 158us/step - loss: 0.6066 - acc: 0.7988 - va
Epoch 9/50
50000/50000 [==============================] - 8s 157us/step - loss: 0.5988 - acc: 0.7996 - va
Epoch 10/50
50000/50000 [==============================] - 8s 159us/step - loss: 0.5929 - acc: 0.8025 - va
Epoch 11/50
50000/50000 [==============================] - 9s 174us/step - loss: 0.6040 - acc: 0.8004 - va
Epoch 12/50
50000/50000 [==============================] - 8s 167us/step - loss: 0.6011 - acc: 0.7978 - va
Epoch 13/50
50000/50000 [==============================] - 8s 159us/step - loss: 0.5947 - acc: 0.8013 - va
Epoch 14/50
50000/50000 [==============================] - 9s 173us/step - loss: 0.6035 - acc: 0.7991 - va
Epoch 15/50
50000/50000 [==============================] - 8s 157us/step - loss: 0.6087 - acc: 0.8009 - va
Epoch 16/50
50000/50000 [==============================] - 8s 156us/step - loss: 0.5955 - acc: 0.8003 - va
Epoch 17/50
50000/50000 [==============================] - 8s 158us/step - loss: 0.5906 - acc: 0.8056 - va
Epoch 18/50
50000/50000 [==============================] - 8s 157us/step - loss: 0.5906 - acc: 0.8059 - va
Epoch 19/50
50000/50000 [==============================] - 8s 158us/step - loss: 0.5963 - acc: 0.8027 - va
Epoch 20/50
50000/50000 [==============================] - 8s 157us/step - loss: 0.5928 - acc: 0.8050 - va
Epoch 21/50
50000/50000 [==============================] - 9s 176us/step - loss: 0.5989 - acc: 0.8028 - va
Epoch 22/50
50000/50000 [==============================] - 8s 165us/step - loss: 0.5789 - acc: 0.8107 - va
Epoch 23/50
50000/50000 [==============================] - 8s 159us/step - loss: 0.5870 - acc: 0.8048 - va
Epoch 24/50
50000/50000 [==============================] - 8s 157us/step - loss: 0.5931 - acc: 0.7998 - va
Epoch 25/50
50000/50000 [==============================] - 8s 159us/step - loss: 0.6003 - acc: 0.7986 - va
Epoch 26/50
```

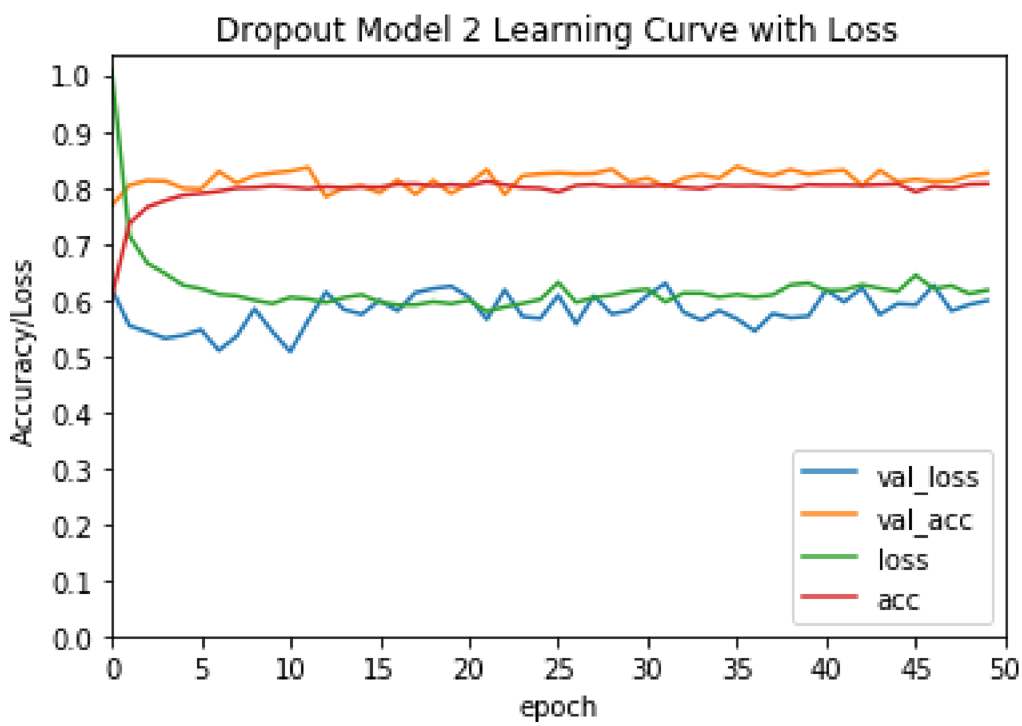
```
50000/50000 [==============================] - 8s 157us/step - loss: 0.6305 - acc: 0.7918 - val
Epoch 27/50
50000/50000 [==============================] - 8s 157us/step - loss: 0.5952 - acc: 0.8034 - val
Epoch 28/50
50000/50000 [==============================] - 8s 158us/step - loss: 0.6034 - acc: 0.8053 - val
Epoch 29/50
50000/50000 [==============================] - 8s 159us/step - loss: 0.6078 - acc: 0.8016 - val
Epoch 30/50
50000/50000 [==============================] - 8s 157us/step - loss: 0.6149 - acc: 0.8025 - val
Epoch 31/50
50000/50000 [==============================] - 9s 174us/step - loss: 0.6187 - acc: 0.8021 - val
Epoch 32/50
50000/50000 [==============================] - 8s 167us/step - loss: 0.5958 - acc: 0.8045 - val
Epoch 33/50
50000/50000 [==============================] - 8s 158us/step - loss: 0.6117 - acc: 0.7998 - val
Epoch 34/50
50000/50000 [==============================] - 8s 158us/step - loss: 0.6114 - acc: 0.7977 - val
Epoch 35/50
50000/50000 [==============================] - 8s 158us/step - loss: 0.6045 - acc: 0.8038 - val
Epoch 36/50
50000/50000 [==============================] - 9s 171us/step - loss: 0.6091 - acc: 0.8027 - val
Epoch 37/50
50000/50000 [==============================] - 9s 177us/step - loss: 0.6045 - acc: 0.8032 - val
Epoch 38/50
50000/50000 [==============================] - 8s 157us/step - loss: 0.6084 - acc: 0.8011 - val
Epoch 39/50
50000/50000 [==============================] - 8s 157us/step - loss: 0.6261 - acc: 0.7992 - val
Epoch 40/50
50000/50000 [==============================] - 8s 158us/step - loss: 0.6295 - acc: 0.8048 - val
Epoch 41/50
50000/50000 [==============================] - 9s 179us/step - loss: 0.6160 - acc: 0.8039 - val
Epoch 42/50
50000/50000 [==============================] - 8s 162us/step - loss: 0.6170 - acc: 0.8042 - val
Epoch 43/50
50000/50000 [==============================] - 8s 158us/step - loss: 0.6265 - acc: 0.8034 - val
Epoch 44/50
50000/50000 [==============================] - 8s 158us/step - loss: 0.6201 - acc: 0.8049 - val
Epoch 45/50
50000/50000 [==============================] - 8s 157us/step - loss: 0.6144 - acc: 0.8062 - val
Epoch 46/50
50000/50000 [==============================] - 8s 158us/step - loss: 0.6433 - acc: 0.7921 - val
Epoch 47/50
50000/50000 [==============================] - 8s 158us/step - loss: 0.6195 - acc: 0.8019 - val
Epoch 48/50
50000/50000 [==============================] - 8s 158us/step - loss: 0.6248 - acc: 0.7988 - val
Epoch 49/50
50000/50000 [==============================] - 8s 157us/step - loss: 0.6105 - acc: 0.8056 - val
Epoch 50/50
```

```
50000/50000 [==============================] - 8s 158us/step - loss: 0.6169 - acc: 0.8065 - va
```

```
In [40]: print('Dropout model 2 test score:',
               model_dropout2.evaluate(X_test, y_test))
```

```
10000/10000 [==============================] - 0s 44us/step
Dropout model 2 test score: [0.6149588119506836, 0.8166]
```

```
In [42]: _ = pd.DataFrame(dropout2.history).plot(
             title = 'Dropout Model 2 Learning Curve with Loss',
             xticks = range(0,51,5),
             yticks = [0.1* x for x in range(0,11)]
         )
         _ = plt.legend(loc = 4)
         _ = plt.xlabel('epoch')
         _ = plt.ylabel('Accuracy/Loss')
```



Quick observation The larger and deeper net with dropout can effectively control the overfitting, as can be seen in this chart. The training and validation scores are much closer together, while the validation loss does not exhibit the steady increase in the later epochs, as was noticed earlier. This generalizability came at a cost in terms of accuracy, though, as the ended around 0.8.

5 Batch Normalization and Residual Connection Model

In [43]:
```
# 1 layer model, which can't adopt res connect in this
# initiate model
model_batch1 = Sequential()

# flatten layer
model_batch1.add(Flatten(input_shape = (28,28)))

# first layer
model_batch1.add(Dense(128,
                input_dim = 784,
                activation='relu'))

# batchnormalize layer
model_batch1.add(BatchNormalization())

# output layer
model_batch1.add(Dense(10, activation='softmax'))

# compile
model_batch1.compile(optimizer = 'adam',
              loss = 'sparse_categorical_crossentropy',
              metrics = ['accuracy'])

model_batch1.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_5 (Flatten)          (None, 784)               0
_________________________________________________________________
dense_19 (Dense)             (None, 128)               100480
_________________________________________________________________
batch_normalization_1 (Batch (None, 128)               512
_________________________________________________________________
dense_20 (Dense)             (None, 10)                1290
=================================================================
Total params: 102,282
Trainable params: 102,026
Non-trainable params: 256
_________________________________________________________________
```

In [44]:
```
batch1 = model_batch1.fit(X_train,
                          y_train,
                          epochs=50,
                          validation_split=10000/60000)
```

```
Train on 50000 samples, validate on 10000 samples
Epoch 1/50
50000/50000 [==============================] - 7s 140us/step - loss: 0.4980 - acc: 0.8264 - val
Epoch 2/50
50000/50000 [==============================] - 6s 122us/step - loss: 0.4037 - acc: 0.8571 - val
Epoch 3/50
50000/50000 [==============================] - 7s 138us/step - loss: 0.3725 - acc: 0.8670 - val
Epoch 4/50
50000/50000 [==============================] - 7s 136us/step - loss: 0.3536 - acc: 0.8733 - val
Epoch 5/50
50000/50000 [==============================] - 6s 124us/step - loss: 0.3396 - acc: 0.8768 - val
Epoch 6/50
50000/50000 [==============================] - 6s 124us/step - loss: 0.3270 - acc: 0.8811 - val
Epoch 7/50
50000/50000 [==============================] - 6s 123us/step - loss: 0.3181 - acc: 0.8822 - val
Epoch 8/50
50000/50000 [==============================] - 7s 146us/step - loss: 0.3093 - acc: 0.8867 - val
Epoch 9/50
50000/50000 [==============================] - 7s 137us/step - loss: 0.3039 - acc: 0.8888 - val
Epoch 10/50
50000/50000 [==============================] - 6s 124us/step - loss: 0.2971 - acc: 0.8907 - val
Epoch 11/50
50000/50000 [==============================] - 6s 124us/step - loss: 0.2915 - acc: 0.8918 - val
Epoch 12/50
50000/50000 [==============================] - 6s 124us/step - loss: 0.2822 - acc: 0.8961 - val
Epoch 13/50
50000/50000 [==============================] - 6s 125us/step - loss: 0.2770 - acc: 0.8986 - val
Epoch 14/50
50000/50000 [==============================] - 6s 124us/step - loss: 0.2715 - acc: 0.9007 - val
Epoch 15/50
50000/50000 [==============================] - 6s 130us/step - loss: 0.2650 - acc: 0.9024 - val
Epoch 16/50
50000/50000 [==============================] - 7s 142us/step - loss: 0.2637 - acc: 0.9019 - val
Epoch 17/50
50000/50000 [==============================] - 6s 127us/step - loss: 0.2604 - acc: 0.9042 - val
Epoch 18/50
50000/50000 [==============================] - 6s 125us/step - loss: 0.2516 - acc: 0.9061 - val
Epoch 19/50
50000/50000 [==============================] - 6s 124us/step - loss: 0.2465 - acc: 0.9081 - val
Epoch 20/50
50000/50000 [==============================] - 6s 125us/step - loss: 0.2487 - acc: 0.9094 - val
Epoch 21/50
50000/50000 [==============================] - 6s 124us/step - loss: 0.2429 - acc: 0.9114 - val
Epoch 22/50
50000/50000 [==============================] - 6s 124us/step - loss: 0.2365 - acc: 0.9117 - val
Epoch 23/50
50000/50000 [==============================] - 6s 125us/step - loss: 0.2337 - acc: 0.9131 - val
Epoch 24/50
```

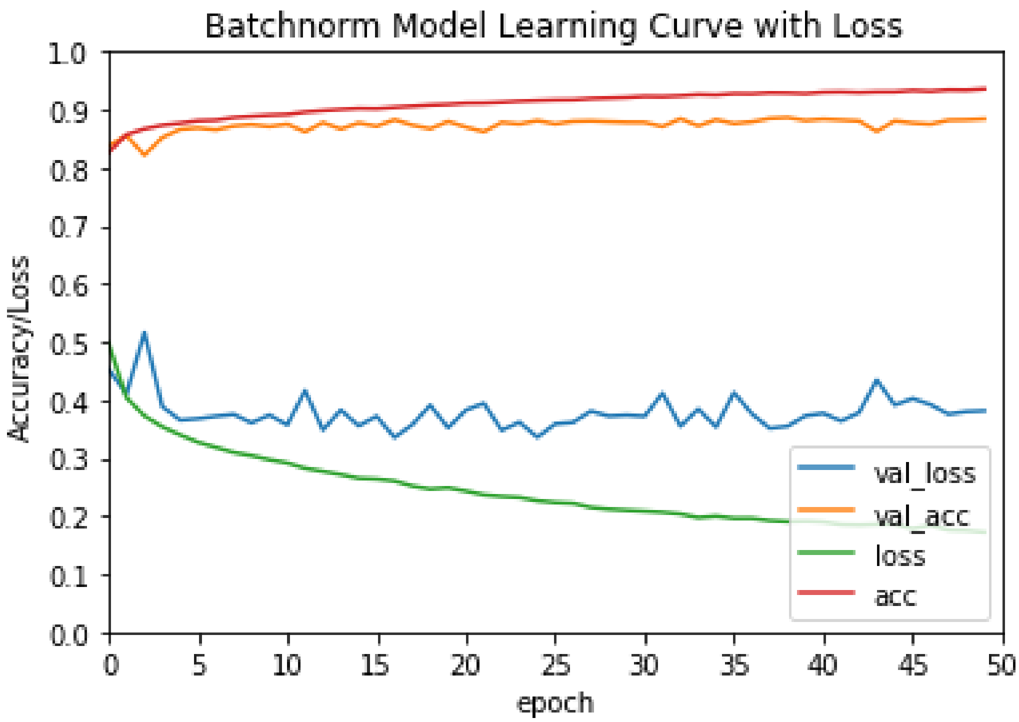
```
50000/50000 [==============================] - 6s 125us/step - loss: 0.2320 - acc: 0.9148 - val
Epoch 25/50
50000/50000 [==============================] - 6s 125us/step - loss: 0.2263 - acc: 0.9161 - val
Epoch 26/50
50000/50000 [==============================] - 7s 130us/step - loss: 0.2233 - acc: 0.9170 - val
Epoch 27/50
50000/50000 [==============================] - 7s 134us/step - loss: 0.2219 - acc: 0.9172 - val
Epoch 28/50
50000/50000 [==============================] - 7s 138us/step - loss: 0.2147 - acc: 0.9193 - val
Epoch 29/50
50000/50000 [==============================] - 7s 136us/step - loss: 0.2117 - acc: 0.9203 - val
Epoch 30/50
50000/50000 [==============================] - 6s 123us/step - loss: 0.2098 - acc: 0.9214 - val
Epoch 31/50
50000/50000 [==============================] - 6s 123us/step - loss: 0.2083 - acc: 0.9232 - val
Epoch 32/50
50000/50000 [==============================] - 6s 124us/step - loss: 0.2064 - acc: 0.9228 - val
Epoch 33/50
50000/50000 [==============================] - 6s 124us/step - loss: 0.2036 - acc: 0.9241 - val
Epoch 34/50
50000/50000 [==============================] - 6s 124us/step - loss: 0.1972 - acc: 0.9263 - val
Epoch 35/50
50000/50000 [==============================] - 6s 123us/step - loss: 0.1998 - acc: 0.9255 - val
Epoch 36/50
50000/50000 [==============================] - 6s 123us/step - loss: 0.1959 - acc: 0.9279 - val
Epoch 37/50
50000/50000 [==============================] - 6s 124us/step - loss: 0.1960 - acc: 0.9276 - val
Epoch 38/50
50000/50000 [==============================] - 6s 123us/step - loss: 0.1920 - acc: 0.9293 - val
Epoch 39/50
50000/50000 [==============================] - 6s 123us/step - loss: 0.1904 - acc: 0.9289 - val
Epoch 40/50
50000/50000 [==============================] - 6s 124us/step - loss: 0.1912 - acc: 0.9279 - val
Epoch 41/50
50000/50000 [==============================] - 7s 141us/step - loss: 0.1898 - acc: 0.9303 - val
Epoch 42/50
50000/50000 [==============================] - 7s 133us/step - loss: 0.1853 - acc: 0.9308 - val
Epoch 43/50
50000/50000 [==============================] - 6s 124us/step - loss: 0.1845 - acc: 0.9298 - val
Epoch 44/50
50000/50000 [==============================] - 6s 123us/step - loss: 0.1856 - acc: 0.9309 - val
Epoch 45/50
50000/50000 [==============================] - 6s 123us/step - loss: 0.1855 - acc: 0.9309 - val
Epoch 46/50
50000/50000 [==============================] - 6s 123us/step - loss: 0.1783 - acc: 0.9331 - val
Epoch 47/50
50000/50000 [==============================] - 6s 124us/step - loss: 0.1839 - acc: 0.9319 - val
Epoch 48/50
```

```
50000/50000 [==============================] - 6s 123us/step - loss: 0.1758 - acc: 0.9341 - val
Epoch 49/50
50000/50000 [==============================] - 6s 125us/step - loss: 0.1748 - acc: 0.9338 - val
Epoch 50/50
50000/50000 [==============================] - 6s 124us/step - loss: 0.1723 - acc: 0.9359 - val
```

```
In [45]: print('Batch normalized model 1 test score:',
               model_batch1.evaluate(X_test, y_test))
```

```
10000/10000 [==============================] - 0s 45us/step
Batch normalized model 1 test score: [0.3997763850092888, 0.8806]
```

```
In [46]: _ = pd.DataFrame(batch1.history).plot(
             title = 'Batchnorm Model Learning Curve with Loss',
             xticks = range(0,51,5),
             yticks = [0.1* x for x in range(0,11)]
         )
         _ = plt.legend(loc = 4)
         _ = plt.xlabel('epoch')
         _ = plt.ylabel('Accuracy/Loss')
```



Larger and deeper dropout batch norm. and residual connection model

```
In [47]: # 6 layer model
         num_class = 10

         # define input layer
         inputs = Input(shape=(28,28,1))

         # flatten
         flat = Flatten()(inputs)

         # 1st Dense layer
         L1_1 = Dense(512, activation='relu')(flat)
         L1_2 = BatchNormalization()(L1_1)

         # 2nd layer
         L2_1 = Dense(512, activation='relu')(L1_2)
         L2_2 = BatchNormalization()(L2_1)

         # 3rd layer
         L3_1 = Dense(512, activation='relu')(L2_2)
         L3_2 = BatchNormalization()(L3_1)

         # skip1
         skip1 = add([L1_2, L3_2])

         # 4th layer
         L4_1 = Dense(512, activation='relu')(skip1)
         L4_2 = BatchNormalization()(L4_1)

         # 5th layer
         L5_1 = Dense(512, activation='relu')(L4_2)
         L5_2 = BatchNormalization()(L5_1)

         #skip2
         skip2 = add([skip1, L5_2])

         # 6th layer
         L6_1 = Dense(512, activation='relu')(skip2)
         L6_2 = BatchNormalization()(L6_1)

         # output layer
         dense = Dense(num_class, activation='softmax')(L6_2)

         # compile
         model_batch2 = Model(inputs = inputs, outputs = dense)
         model_batch2.compile(optimizer = 'adam',
                       loss = 'sparse_categorical_crossentropy',
```

```
                    metrics = ['accuracy'])
        model_batch2.summary()
```

```
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
input_1 (InputLayer)             (None, 28, 28, 1)     0
____________________________________________________________________________________________________
flatten_6 (Flatten)              (None, 784)           0           input_1[0][0]
____________________________________________________________________________________________________
dense_21 (Dense)                 (None, 512)           401920      flatten_6[0][0]
____________________________________________________________________________________________________
batch_normalization_2 (BatchNor (None, 512)           2048        dense_21[0][0]
____________________________________________________________________________________________________
dense_22 (Dense)                 (None, 512)           262656      batch_normalization_2[0][0]
____________________________________________________________________________________________________
batch_normalization_3 (BatchNor (None, 512)           2048        dense_22[0][0]
____________________________________________________________________________________________________
dense_23 (Dense)                 (None, 512)           262656      batch_normalization_3[0][0]
____________________________________________________________________________________________________
batch_normalization_4 (BatchNor (None, 512)           2048        dense_23[0][0]
____________________________________________________________________________________________________
add_1 (Add)                      (None, 512)           0           batch_normalization_2[0][0]
                                                                   batch_normalization_4[0][0]
____________________________________________________________________________________________________
dense_24 (Dense)                 (None, 512)           262656      add_1[0][0]
____________________________________________________________________________________________________
batch_normalization_5 (BatchNor (None, 512)           2048        dense_24[0][0]
____________________________________________________________________________________________________
dense_25 (Dense)                 (None, 512)           262656      batch_normalization_5[0][0]
____________________________________________________________________________________________________
batch_normalization_6 (BatchNor (None, 512)           2048        dense_25[0][0]
____________________________________________________________________________________________________
add_2 (Add)                      (None, 512)           0           add_1[0][0]
                                                                   batch_normalization_6[0][0]
____________________________________________________________________________________________________
dense_26 (Dense)                 (None, 512)           262656      add_2[0][0]
____________________________________________________________________________________________________
batch_normalization_7 (BatchNor (None, 512)           2048        dense_26[0][0]
____________________________________________________________________________________________________
dense_27 (Dense)                 (None, 10)            5130        batch_normalization_7[0][0]
====================================================================================================
Total params: 1,732,618
Trainable params: 1,726,474
Non-trainable params: 6,144

____________________________________________________________________________________________________
```

```
In [49]: X_train_ = X_train.reshape(X_train.shape[0], 28, 28, 1)
```

```
batch2 = model_batch2.fit(X_train_,
                          y_train,
                          epochs=50,
                          validation_split=10000/60000)
```

```
Train on 50000 samples, validate on 10000 samples
Epoch 1/50
50000/50000 [==============================] - 20s 401us/step - loss: 0.5506 - acc: 0.8056 - va
Epoch 2/50
50000/50000 [==============================] - 18s 366us/step - loss: 0.4273 - acc: 0.8462 - va
Epoch 3/50
50000/50000 [==============================] - 19s 379us/step - loss: 0.3911 - acc: 0.8565 - va
Epoch 4/50
50000/50000 [==============================] - 19s 386us/step - loss: 0.3614 - acc: 0.8686 - va
Epoch 5/50
50000/50000 [==============================] - 19s 371us/step - loss: 0.3401 - acc: 0.8759 - va
Epoch 6/50
50000/50000 [==============================] - 18s 364us/step - loss: 0.3203 - acc: 0.8821 - va
Epoch 7/50
50000/50000 [==============================] - 18s 364us/step - loss: 0.3081 - acc: 0.8872 - va
Epoch 8/50
50000/50000 [==============================] - 18s 368us/step - loss: 0.2918 - acc: 0.8905 - va
Epoch 9/50
50000/50000 [==============================] - 19s 387us/step - loss: 0.2831 - acc: 0.8946 - va
Epoch 10/50
50000/50000 [==============================] - 18s 363us/step - loss: 0.2708 - acc: 0.8992 - va
Epoch 11/50
50000/50000 [==============================] - 18s 363us/step - loss: 0.2619 - acc: 0.9020 - va
Epoch 12/50
50000/50000 [==============================] - 18s 365us/step - loss: 0.2472 - acc: 0.9075 - va
Epoch 13/50
50000/50000 [==============================] - 20s 406us/step - loss: 0.2409 - acc: 0.9093 - va
Epoch 14/50
50000/50000 [==============================] - 19s 378us/step - loss: 0.2339 - acc: 0.9127 - va
Epoch 15/50
50000/50000 [==============================] - 18s 366us/step - loss: 0.2265 - acc: 0.9151 - va
Epoch 16/50
50000/50000 [==============================] - 18s 362us/step - loss: 0.2188 - acc: 0.9164 - va
Epoch 17/50
50000/50000 [==============================] - 19s 388us/step - loss: 0.2128 - acc: 0.9187 - va
Epoch 18/50
50000/50000 [==============================] - 18s 366us/step - loss: 0.2064 - acc: 0.9202 - va
Epoch 19/50
50000/50000 [==============================] - 19s 379us/step - loss: 0.1995 - acc: 0.9225 - va
Epoch 20/50
50000/50000 [==============================] - 18s 367us/step - loss: 0.1909 - acc: 0.9280 - va
Epoch 21/50
50000/50000 [==============================] - 19s 373us/step - loss: 0.1879 - acc: 0.9276 - va
```

```
Epoch 22/50
50000/50000 [==============================] - 19s 384us/step - loss: 0.1804 - acc: 0.9302 - va
Epoch 23/50
50000/50000 [==============================] - 18s 365us/step - loss: 0.1736 - acc: 0.9335 - va
Epoch 24/50
50000/50000 [==============================] - 18s 367us/step - loss: 0.1692 - acc: 0.9345 - va
Epoch 25/50
50000/50000 [==============================] - 18s 366us/step - loss: 0.1658 - acc: 0.9371 - va
Epoch 26/50
50000/50000 [==============================] - 20s 390us/step - loss: 0.1613 - acc: 0.9383 - va
Epoch 27/50
50000/50000 [==============================] - 18s 368us/step - loss: 0.1568 - acc: 0.9396 - va
Epoch 28/50
50000/50000 [==============================] - 18s 366us/step - loss: 0.1513 - acc: 0.9411 - va
Epoch 29/50
50000/50000 [==============================] - 18s 366us/step - loss: 0.1450 - acc: 0.9439 - va
Epoch 30/50
50000/50000 [==============================] - 20s 402us/step - loss: 0.1397 - acc: 0.9456 - va
Epoch 31/50
50000/50000 [==============================] - 18s 365us/step - loss: 0.1401 - acc: 0.9471 - va
Epoch 32/50
50000/50000 [==============================] - 18s 369us/step - loss: 0.1362 - acc: 0.9474 - va
Epoch 33/50
50000/50000 [==============================] - 18s 365us/step - loss: 0.1322 - acc: 0.9494 - va
Epoch 34/50
50000/50000 [==============================] - 19s 378us/step - loss: 0.1272 - acc: 0.9510 - va
Epoch 35/50
50000/50000 [==============================] - 19s 388us/step - loss: 0.1251 - acc: 0.9523 - va
Epoch 36/50
50000/50000 [==============================] - 18s 370us/step - loss: 0.1200 - acc: 0.9539 - va
Epoch 37/50
50000/50000 [==============================] - 18s 367us/step - loss: 0.1203 - acc: 0.9534 - va
Epoch 38/50
50000/50000 [==============================] - 18s 367us/step - loss: 0.1145 - acc: 0.9553 - va
Epoch 39/50
50000/50000 [==============================] - 20s 391us/step - loss: 0.1118 - acc: 0.9562 - va
Epoch 40/50
50000/50000 [==============================] - 18s 364us/step - loss: 0.1065 - acc: 0.9589 - va
Epoch 41/50
50000/50000 [==============================] - 18s 363us/step - loss: 0.1065 - acc: 0.9583 - va
Epoch 42/50
50000/50000 [==============================] - 18s 366us/step - loss: 0.1062 - acc: 0.9593 - va
Epoch 43/50
50000/50000 [==============================] - 20s 390us/step - loss: 0.1022 - acc: 0.9607 - va
Epoch 44/50
50000/50000 [==============================] - 18s 365us/step - loss: 0.1026 - acc: 0.9606 - va
Epoch 45/50
50000/50000 [==============================] - 18s 366us/step - loss: 0.0975 - acc: 0.9622 - va
```

```
Epoch 46/50
50000/50000 [==============================] - 18s 364us/step - loss: 0.0945 - acc: 0.9631 - va
Epoch 47/50
50000/50000 [==============================] - 21s 414us/step - loss: 0.0935 - acc: 0.9641 - va
Epoch 48/50
50000/50000 [==============================] - 19s 374us/step - loss: 0.0889 - acc: 0.9661 - va
Epoch 49/50
50000/50000 [==============================] - 18s 365us/step - loss: 0.0861 - acc: 0.9674 - va
Epoch 50/50
50000/50000 [==============================] - 18s 366us/step - loss: 0.0883 - acc: 0.9662 - va
```

```
In [50]: X_test_ = X_test.reshape(X_test.shape[0], 28, 28, 1)
         print('Batch normalized model 2 test score:',
               model_batch2.evaluate(X_test_, y_test))
```

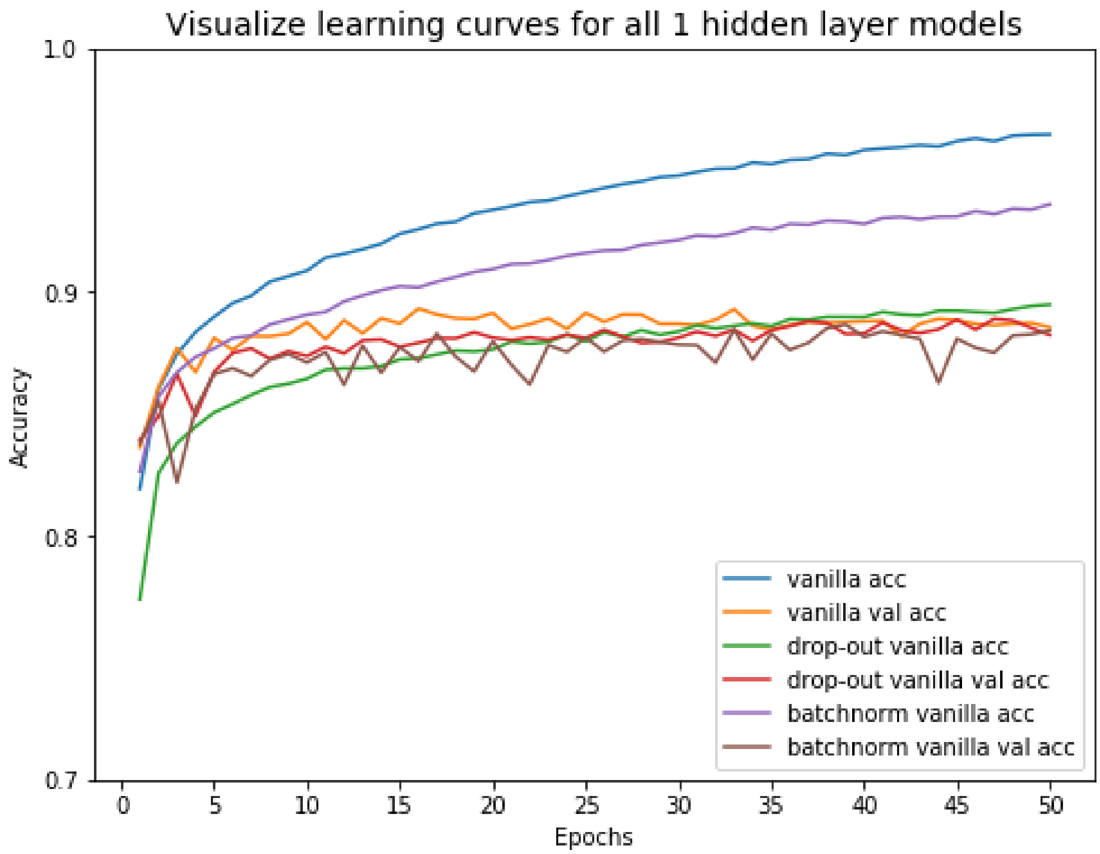
```
10000/10000 [==============================] - 1s 76us/step
Batch normalized model 2 test score: [0.4340573483712971, 0.8855]
```

```
In [51]: _ = pd.DataFrame(batch2.history).plot(
             title = 'ResNet Batchnorm Model Learning Curve with Loss',
             xticks = range(0,51,5),
             yticks = [0.1* x for x in range(0,11)]
         )
         _ = plt.legend(loc = 4)
         _ = plt.xlabel('epoch')
         _ = plt.ylabel('Accuracy/Loss')
```

6 Visualization

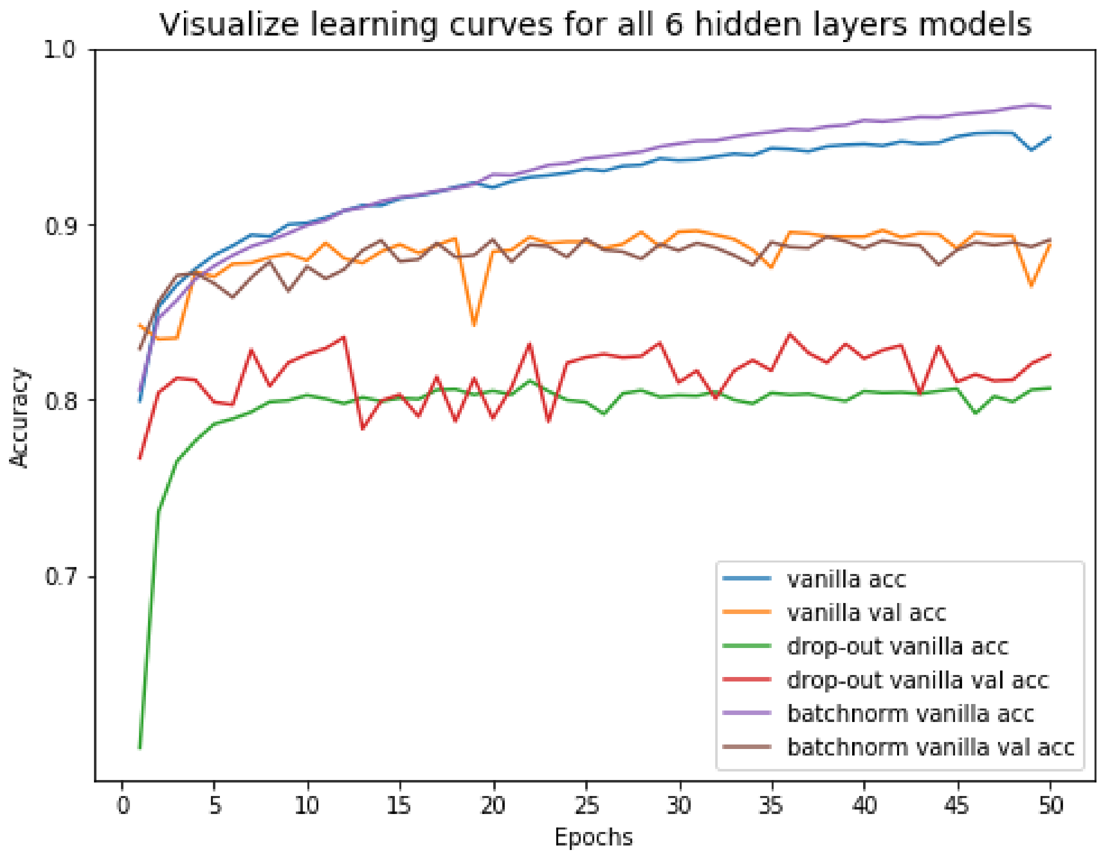
```
In [61]: plt.figure(figsize=(8, 6))
         _ = sns.lineplot(y = vanilla1.history['acc'],
                      x = range(1,51), label='vanilla acc')
         _ = sns.lineplot(y = vanilla1.history['val_acc'],
                      x = range(1,51),label='vanilla val acc')
         _ = sns.lineplot(y = dropout1.history['acc'],
                      x = range(1,51),label='drop-out vanilla acc')
         _ = sns.lineplot(y = dropout1.history['val_acc'],
                      x = range(1,51),label='drop-out vanilla val acc')
         _ = sns.lineplot(y = batch1.history['acc'],
                      x = range(1,51),label='batchnorm vanilla acc')
         _ = sns.lineplot(y = batch1.history['val_acc'],
                      x = range(1,51),label='batchnorm vanilla val acc')

         _ = plt.title('Visualize learning curves for all 1 hidden layer models',size=14)
         _ = plt.xlabel('Epochs')
         _ = plt.ylabel('Accuracy')
         _ = plt.yticks([x/10 for x in range(7,11)])
         _ = plt.xticks(range(0,51,5))
```



In [64]:
```
plt.figure(figsize=(8, 6))
_ = sns.lineplot(y = vanilla2.history['acc'],
                 x = range(1,51), label='vanilla acc')
_ = sns.lineplot(y = vanilla2.history['val_acc'],
                 x = range(1,51),label='vanilla val acc')
_ = sns.lineplot(y = dropout2.history['acc'],
                 x = range(1,51),label='drop-out vanilla acc')
_ = sns.lineplot(y = dropout2.history['val_acc'],
                 x = range(1,51),label='drop-out vanilla val acc')
_ = sns.lineplot(y = batch2.history['acc'],
                 x = range(1,51),label='batchnorm vanilla acc')
_ = sns.lineplot(y = batch2.history['val_acc'],
                 x = range(1,51),label='batchnorm vanilla val acc')

_ = plt.title('Visualize learning curves for all 6 hidden layers models',size=14)
_ = plt.xlabel('Epochs')
_ = plt.ylabel('Accuracy')
_ = plt.yticks([x/10 for x in range(7,11)])
_ = plt.xticks(range(0,51,5))
```



7 Summary

In this task, we tested six models: a shallow (1 hidden layer) and a deeper (6 hidden layers) model for three model types:

1. Base model
2. Model with dropout
3. Model with batch normalization and residual connections (without dropout)

Our most successful model, in terms of validation set accuracy, was not surprisingly the deeper model with six hidden layers and batch normalization (without dropout). It approached the mid-to-upper 80s in accuracy. The vanilla model was nearly identical in validation set accuracy, and slightly below the training set accuracy of the batch normalization model.

Drop out did not seem to be an effetcive strategy, as accuracy scores were considerably below those of the base model and the batch normalization model. It should be noted, however, that it was very effective at reducing the degree of overfitting, as the training and validation scores were nearly identical.