

jupyter 20191024HW\_2\_MNIST CNN Last Checkpoint: 撥移前 (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Not Trusted Python 3

```
In [0]: 1 import pandas as pd
2 import numpy as np
3 from sklearn.model_selection import train_test_split
4 from keras.utils import to_categorical
5 from keras.models import Sequential
6 from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D, AvgPool2D, BatchNormalization, Reshape
7 from keras.preprocessing.image import ImageDataGenerator
8 from keras.callbacks import LearningRateScheduler
9 import matplotlib.pyplot as plt
```

```
In [45]: 1 # Colabatory 使用雲端硬碟檔案：安裝 pydrive 條件
2 # 取得檔案共用連接埠 → 請在「10」後面的「1_wVrImzehPbIXaDDkfqebNg0Yc18pl3」就是檔案連結碼。
3 from pydrive.auth import GoogleAuth
4 from pydrive.drive import GoogleDrive
5 from google.colab import auth
6 from oauth2client.client import GoogleCredentials
7 auth.authenticate_user()
8 gauth = GoogleAuth()
9 gauth.credentials = GoogleCredentials.get_application_default()
10 drive = GoogleDrive(gauth)
11 file_id = '1RAEXRs036IpKHToxB8SrnfBfthNLwMwQ' #雲端硬碟檔案連結碼
12 downloaded = drive.CreateFile({'id': file_id})
13 downloaded.GetContentFile('MNIST_train.csv')
14 train = pd.read_csv("MNIST_train.csv")
15 print(train)

WARNING:googleapiclient.discovery_cache:file_cache is unavailable when using oauth2client >= 4.0.0 or google-auth
Traceback (most recent call last):
  File "/usr/local/lib/python3.6/dist-packages/googleapiclient/discovery_cache/_init__.py", line 36, in autodetect
    from google.appengine.api import memcache
ModuleNotFoundError: No module named 'google.appengine'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/usr/local/lib/python3.6/dist-packages/googleapiclient/discovery_cache/file_cache.py", line 33, in <module>
    from oauth2client.contrib.locked_file import LockedFile
ModuleNotFoundError: No module named 'oauth2client.contrib.locked_file'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/usr/local/lib/python3.6/dist-packages/googleapiclient/discovery_cache/file_cache.py", line 37, in <module>
    from oauth2client.locked_file import LockedFile
ModuleNotFoundError: No module named 'oauth2client.locked_file'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/usr/local/lib/python3.6/dist-packages/googleapiclient/discovery_cache/_init__.py", line 41, in autodetect
    from . import file_cache
  File "/usr/local/lib/python3.6/dist-packages/googleapiclient/discovery_cache/file_cache.py", line 41, in <module>
    'file_cache is unavailable when using oauth2client >= 4.0.0 or google-auth'
ImportError: file_cache is unavailable when using oauth2client >= 4.0.0 or google-auth

label pixel0 pixel1 pixel2 ... pixel780 pixel781 pixel782 pixel783
0 1 0 0 0 ... 0 0 0 0
1 0 0 0 0 ... 0 0 0 0
2 1 0 0 0 0 ... 0 0 0 0
3 4 0 0 0 0 ... 0 0 0 0
4 0 0 0 0 0 ... 0 0 0 0
...
41995 0 0 0 0 0 ... 0 0 0 0
41996 1 0 0 0 0 ... 0 0 0 0
41997 7 0 0 0 0 ... 0 0 0 0
41998 6 0 0 0 0 ... 0 0 0 0
41999 9 0 0 0 0 ... 0 0 0 0

[42000 rows x 785 columns]
```

```
In [46]: 1 from pydrive.auth import GoogleAuth
2 from pydrive.drive import GoogleDrive
3 from google.colab import auth
4 from oauth2client.client import GoogleCredentials
5 auth.authenticate_user()
6 gauth = GoogleAuth()
7 gauth.credentials = GoogleCredentials.get_application_default()
8 drive = GoogleDrive(gauth)
9 file_id = '1GNwH5zCeOglnSDd-ObTrQS0HE1b0Gis' #雲端硬碟檔案連結碼
10 downloaded = drive.CreateFile({'id': file_id})
11 downloaded.GetContentFile('MNIST_test.csv')
12 test = pd.read_csv("MNIST_test.csv")
13 print(test)

WARNING:googleapiclient.discovery_cache:file_cache is unavailable when using oauth2client >= 4.0.0 or google-auth
Traceback (most recent call last):
  File "/usr/local/lib/python3.6/dist-packages/googleapiclient/discovery_cache/_init__.py", line 36, in autodetect
    from google.appengine.api import memcache
ModuleNotFoundError: No module named 'google.appengine'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/usr/local/lib/python3.6/dist-packages/googleapiclient/discovery_cache/file_cache.py", line 33, in <module>
    from oauth2client.contrib.locked_file import LockedFile
ModuleNotFoundError: No module named 'oauth2client.contrib.locked_file'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/usr/local/lib/python3.6/dist-packages/googleapiclient/discovery_cache/_init__.py", line 37, in <module>
    from oauth2client.locked_file import LockedFile
ModuleNotFoundError: No module named 'oauth2client.locked_file'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/usr/local/lib/python3.6/dist-packages/googleapiclient/discovery_cache/file_cache.py", line 37, in <module>
    from oauth2client.locked_file import LockedFile
ModuleNotFoundError: No module named 'oauth2client.locked_file'

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/usr/local/lib/python3.6/dist-packages/googleapiclient/discovery_cache/_init__.py", line 41, in autodetect
    from . import file_cache
  File "/usr/local/lib/python3.6/dist-packages/googleapiclient/discovery_cache/file_cache.py", line 41, in <module>
    'file_cache is unavailable when using oauth2client >= 4.0.0 or google-auth'
ImportError: file_cache is unavailable when using oauth2client >= 4.0.0 or google-auth

pixel0 pixel1 pixel2 pixel3 ... pixel780 pixel781 pixel782 pixel783
0 0 0 0 0 ... 0 0 0 0
1 0 0 0 0 0 ... 0 0 0 0
2 0 0 0 0 0 ... 0 0 0 0
3 0 0 0 0 0 ... 0 0 0 0
4 0 0 0 0 0 ... 0 0 0 0
...
27995 0 0 0 0 0 ... 0 0 0 0
27996 0 0 0 0 0 ... 0 0 0 0
27997 0 0 0 0 0 ... 0 0 0 0
27998 0 0 0 0 0 ... 0 0 0 0
27999 0 0 0 0 0 ... 0 0 0 0

[28000 rows x 784 columns]
```

```
In [0]: 1 train = pd.read_csv("MNIST_train.csv")
2 test = pd.read_csv("MNIST_test.csv")
3
4 Y_train = train["label"]
5 X_train = train.drop(labels = ["label"],axis = 1)
6 X_train = X_train / 255.0
7 X_test = test / 255.0
8 X_train = X_train.values.reshape(-1,28,28,1)
9 X_test = X_test.values.reshape(-1,28,28,1)
10 Y_train = to_categorical(Y_train, num_classes = 10)
```

```
In [0]: 1 annealer = LearningRateScheduler(lambda x: 1e-3 * 0.95 ** x, verbose=0)
2 styles=[':', '-.', '--', ':', '-.', '--', ':', '-.', '--', '-']
```

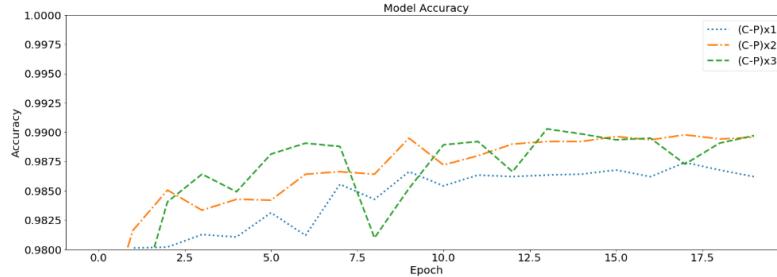
## Convolution-Subsampling Pairs

```
In [0]: 1 # 逐行審視模型
2 nets = 3
3 model = [0] *nets
4
5 for j in range(3):
6     model[j] = Sequential()
7     model[j].add(Conv2D(24,kernel_size=5,padding='same',activation='relu',
8     input_shape=(28,28,1)))
9     model[j].add(MaxPool2D())
10    if j>0:
11        model[j].add(Conv2D(48,kernel_size=5,padding='same',activation='relu'))
12        model[j].add(MaxPool2D())
13    if j>1:
14        model[j].add(Conv2D(64,kernel_size=5,padding='same',activation='relu'))
15        model[j].add(Flatten())
16    model[j].add(Dense(256, activation='relu'))
17    model[j].add(Dense(10, activation='softmax'))
18    model[j].compile(optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"])

In [50]: 1 # 訓練與驗證
2 X_train2, X_val2, Y_train2, Y_val2 = train_test_split(X_train, Y_train, test_size = 0.333)
3 # Train & Validation
4 history = [0] * nets
5 names = ["(C-P)x1","(C-P)x2","(C-P)x3"]
6 epochs = 20
7 for j in range(nets):
8     history[j] = model[j].fit(X_train2,Y_train2, batch_size=80, epochs = epochs,
9     validation_data = (X_val2,Y_val2), callbacks=[annealer], verbose=0)
10    print("CNN (%d): Epochs=%d, Train accuracy=%f, Validation accuracy=%f" %format(
11        names[j],epochs,max(history[j].history['acc']),max(history[j].history['val_acc'])))

CNN (C-P)x1: Epochs=20, Train accuracy=0.99996, Validation accuracy=0.99742
CNN (C-P)x2: Epochs=20, Train accuracy=0.99996, Validation accuracy=0.98978
CNN (C-P)x3: Epochs=20, Train accuracy=0.99986, Validation accuracy=0.99028
```

```
In [51]: 1 # 繪製訓練曲線
2 plt.figure(figsize=(24,8))
3 for i in range(nets):
4     plt.plot(history[i].history['val_acc'],linestyle=styles[i],linewidth=3)
5     plt.title('Model Accuracy',fontsize=20)
6     plt.xlabel('Accuracy',fontsize=20)
7     plt.ylabel('Epoch',fontsize=20)
8     plt.legend(names, loc='best',fontsize=20)
9     plt.xticks(fontsize=20)
10    plt.yticks(fontsize=20)
11    axes = plt.gca()
12    axes.set_ylim([0.98,1])
13    plt.savefig("MNIST_CNN (C-P).jpg", dpi=300)
14    plt.show()
```



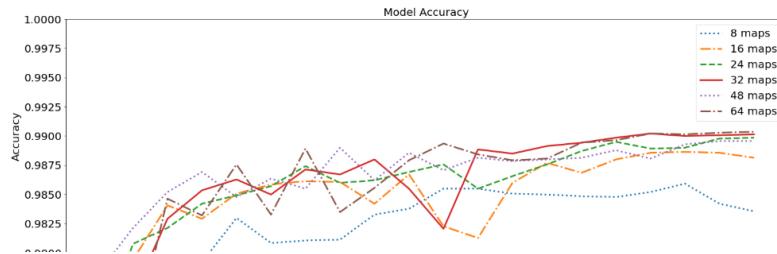
## Feature Maps

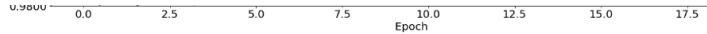
```
In [0]: 1 nets = 6
2 model = [0] *nets
3 for j in range(6):
4     model[j] = Sequential()
5     model[j].add(Conv2D(j*8, kernel_size=5,activation='relu',input_shape=(28,28,1)))
6     model[j].add(MaxPool2D())
7     model[j].add(Conv2D(j*16,16,kernel_size=5,activation='relu'))
8     model[j].add(MaxPool2D())
9     model[j].add(Flatten())
10    model[j].add(Dense(256, activation='relu'))
11    model[j].add(Dense(10, activation='softmax'))
12    model[j].compile(optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"])

In [53]: 1 X_train2, X_val2, Y_train2, Y_val2 = train_test_split(X_train, Y_train, test_size = 0.333)
2 history = [0] * nets
3 names = ["8 maps","16 maps","24 maps","32 maps","48 maps","64 maps"]
4 epochs = 20
5 for j in range(nets):
6     history[j] = model[j].fit(X_train2,Y_train2, batch_size=80, epochs = epochs,
7     validation_data = (X_val2,Y_val2), callbacks=[annealer], verbose=0)
8     print("CNN %d maps: Epochs=%d, Train accuracy=%f, Validation accuracy=%f" %format(
9         names[j],epochs,max(history[j].history['acc']),max(history[j].history['val_acc'])))

CNN 8 maps: Epochs=20, Train accuracy=0.99946, Validation accuracy=0.98591
CNN 16 maps: Epochs=20, Train accuracy=1.00000, Validation accuracy=0.98983
CNN 24 maps: Epochs=20, Train accuracy=0.99996, Validation accuracy=0.99985
CNN 32 maps: Epochs=20, Train accuracy=0.99996, Validation accuracy=0.99020
CNN 48 maps: Epochs=20, Train accuracy=0.99996, Validation accuracy=0.98956
CNN 64 maps: Epochs=20, Train accuracy=0.99996, Validation accuracy=0.99035
```

```
In [54]: 1 plt.figure(figsize=(24,8))
2 for i in range(nets):
3     plt.plot(history[i].history['val_acc'],linestyle=styles[i],linewidth=3)
4     plt.title('Model Accuracy',fontsize=20)
5     plt.xlabel('Accuracy',fontsize=20)
6     plt.ylabel('Epoch',fontsize=20)
7     plt.legend(names, loc='best',fontsize=20)
8     plt.xticks(fontsize=20)
9     plt.yticks(fontsize=20)
10    axes = plt.gca()
11    axes.set_ylim([0.98,1])
12    plt.savefig("MNIST_CNN (maps).jpg", dpi=300)
13    plt.show()
```





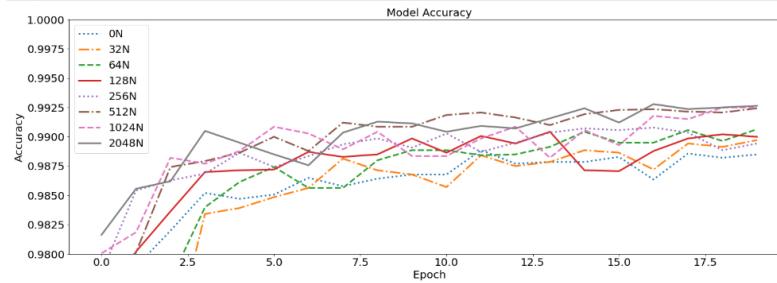
## Dense Layer

```
In [8]:  
1 nets = 8  
2 model = [0] *nets  
3  
4 for j in range(8):  
5     model[j] = Sequential()  
6     model[j].add(Conv2D(32,kernel_size=5,activation='relu',input_shape=(28,28,1)))  
7     model[j].add(MaxPool2D())  
8     model[j].add(Conv2D(64,kernel_size=5,activation='relu'))  
9     model[j].add(MaxPool2D())  
10    model[j].add(Flatten())  
11    if j > 0:  
12        model[j].add(Dense(2**((j+4), activation='relu')))  
13    model[j].add(Dense(10, activation='softmax'))  
14    model[j].compile(optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"])
```

```
In [56]:  
1 X_train2, X_val2, Y_train2, Y_val2 = train_test_split(X_train, Y_train, test_size = 0.333)  
2 history = [0] * nets  
3 names = ["0N","32N","64N","128N","256N","512N","1024N","2048N"]  
4 epochs = 20  
5 for j in range(nets):  
6     history[j] = model[j].fit(X_train2,Y_train2, batch_size=80, epochs = epochs,  
7     validation_data = (X_val2,Y_val2), callbacks=[annealer], verbose=0)  
8     print("CNN (%d): Epochs=%d, Train accuracy=%2.5f, Validation accuracy=%2.5f".format(  
9         names[j],epochs,max(history[j].history['acc']),max(history[j].history['val_acc'])))
```

CNN 0N: Epochs=20, Train accuracy=0.99993, Validation accuracy=0.99885  
CNN 32N: Epochs=20, Train accuracy=0.99982, Validation accuracy=0.99870  
CNN 64N: Epochs=20, Train accuracy=0.99986, Validation accuracy=0.99063  
CNN 128N: Epochs=20, Train accuracy=0.99996, Validation accuracy=0.99042  
CNN 256N: Epochs=20, Train accuracy=0.99996, Validation accuracy=0.99078  
CNN 512N: Epochs=20, Train accuracy=0.99996, Validation accuracy=0.99242  
CNN 1024N: Epochs=20, Train accuracy=0.99996, Validation accuracy=0.99249  
CNN 2048N: Epochs=20, Train accuracy=0.99996, Validation accuracy=0.99278

```
In [57]:  
1 plt.figure(figsize=(24,8))  
2 for i in range(nets):  
3     plt.plot(history[i].history['val_acc'],linestyle=styles[i],linewidth=3)  
4     plt.title('Model Accuracy',fontsize=20)  
5     plt.ylabel('Accuracy',fontsize=20)  
6     plt.xlabel('Epoch',fontsize=20)  
7     plt.legend(names, loc='best',fontsize=20)  
8     plt.xticks(fontsize=20)  
9     plt.yticks(fontsize=20)  
10    axes = plt.gca()  
11    axes.set_ylim([0.98,1])  
12    plt.savefig("MNIST_CNN (OS).jpg", dpi=300)  
13    plt.show()
```



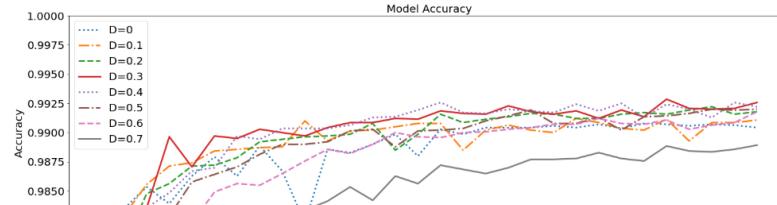
## Dropout

```
In [8]:  
1 nets = 8  
2 model = [0] *nets  
3  
4 for j in range(8):  
5     model[j] = Sequential()  
6     model[j].add(Conv2D(32,kernel_size=5,activation='relu',input_shape=(28,28,1)))  
7     model[j].add(MaxPool2D())  
8     model[j].add(Dropout(j*.1))  
9     model[j].add(Conv2D(64,kernel_size=5,activation='relu'))  
10    model[j].add(MaxPool2D())  
11    model[j].add(Dropout(j*.1))  
12    model[j].add(Flatten())  
13    model[j].add(Dense(128, activation='relu'))  
14    model[j].add(Dropout(j*.1))  
15    model[j].add(Dense(10, activation='softmax'))  
16    model[j].compile(optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"])
```

```
In [59]:  
1 X_train2, X_val2, Y_train2, Y_val2 = train_test_split(X_train, Y_train, test_size = 0.333)  
2 history = [0] * nets  
3 names = ["D=0","D=0.1","D=0.2","D=0.3","D=0.4","D=0.5","D=0.6","D=0.7"]  
4 epochs = 30  
5 for j in range(nets):  
6     history[j] = model[j].fit(X_train2,Y_train2, batch_size=80, epochs = epochs,  
7     validation_data = (X_val2,Y_val2), callbacks=[annealer], verbose=0)  
8     print("CNN (%d): Epochs=%d, Train accuracy=%2.5f, Validation accuracy=%2.5f".format(  
9         names[j],epochs,max(history[j].history['acc']),max(history[j].history['val_acc'])))
```

CNN D=0: Epochs=30, Train accuracy=1.00000, Validation accuracy=0.99078  
CNN D=0.1: Epochs=30, Train accuracy=0.99971, Validation accuracy=0.99221  
CNN D=0.2: Epochs=30, Train accuracy=0.99864, Validation accuracy=0.99221  
CNN D=0.3: Epochs=30, Train accuracy=0.99720, Validation accuracy=0.99285  
CNN D=0.4: Epochs=30, Train accuracy=0.99390, Validation accuracy=0.99256  
CNN D=0.5: Epochs=30, Train accuracy=0.98951, Validation accuracy=0.99199  
CNN D=0.6: Epochs=30, Train accuracy=0.98147, Validation accuracy=0.99178  
CNN D=0.7: Epochs=30, Train accuracy=0.96455, Validation accuracy=0.98892

```
In [60]:  
1 plt.figure(figsize=(24,8))  
2 for i in range(nets):  
3     plt.plot(history[i].history['val_acc'],linestyle=styles[i],linewidth=3)  
4     plt.title('Model Accuracy',fontsize=20)  
5     plt.ylabel('Accuracy',fontsize=20)  
6     plt.xlabel('Epoch',fontsize=20)  
7     plt.legend(names, loc='best',fontsize=20)  
8     plt.xticks(fontsize=20)  
9     plt.yticks(fontsize=20)  
10    axes = plt.gca()  
11    axes.set_ylim([0.98,1])  
12    plt.savefig("MNIST_CNN (drop).jpg", dpi=300)  
13    plt.show()
```





## Advanced Features

```
In [8]: 1 nets = 5
2 model = [0] *nets
3
4 j=0
5 model[j] = Sequential()
6 model[j].add(Conv2D(32,kernel_size=5,activation='relu',input_shape=(28,28,1)))
7 model[j].add(MaxPool2D())
8 model[j].add(Dropout(0.4))
9 model[j].add(Conv2D(64,kernel_size=5,activation='relu'))
10 model[j].add(MaxPool2D())
11 model[j].add(Dropout(0.4))
12 model[j].add(Flatten())
13 model[j].add(Dense(128, activation='relu'))
14 model[j].add(Dropout(0.4))
15 model[j].add(Dense(10, activation='softmax'))
16 model[j].compile(optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"])

In [8]: 1 j=1
2 model[j] = Sequential()
3 model[j].add(Conv2D(32,kernel_size=3,activation='relu',input_shape=(28,28,1)))
4 model[j].add(Conv2D(32,kernel_size=3,activation='relu'))
5 model[j].add(MaxPool2D())
6 model[j].add(Dropout(0.4))
7 model[j].add(Conv2D(64,kernel_size=3,activation='relu'))
8 model[j].add(Conv2D(64,kernel_size=3,activation='relu'))
9 model[j].add(MaxPool2D())
10 model[j].add(Dropout(0.4))
11 model[j].add(Flatten())
12 model[j].add(Dense(128, activation='relu'))
13 model[j].add(Dropout(0.4))
14 model[j].add(Dense(10, activation='softmax'))
15 model[j].compile(optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"])

In [8]: 1 j=2
2 model[j] = Sequential()
3 model[j].add(Conv2D(32,kernel_size=5,activation='relu',input_shape=(28,28,1)))
4 model[j].add(Conv2D(32,kernel_size=5,strides=2,padding='same',activation='relu'))
5 model[j].add(Dropout(0.4))
6 model[j].add(Conv2D(64,kernel_size=5,activation='relu'))
7 model[j].add(Conv2D(64,kernel_size=5,strides=2,padding='same',activation='relu'))
8 model[j].add(Dropout(0.4))
9 model[j].add(Flatten())
10 model[j].add(Dense(128, activation='relu'))
11 model[j].add(Dropout(0.4))
12 model[j].add(Dense(10, activation='softmax'))
13 model[j].compile(optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"])

In [8]: 1 j=3
2 model[j] = Sequential()
3 model[j].add(Conv2D(32,kernel_size=3,activation='relu',input_shape=(28,28,1)))
4 model[j].add(BatchNormalization())
5 model[j].add(Conv2D(32,kernel_size=3,activation='relu'))
6 model[j].add(BatchNormalization())
7 model[j].add(Conv2D(32,kernel_size=5,strides=2,padding='same',activation='relu'))
8 model[j].add(BatchNormalization())
9 model[j].add(Dropout(0.4))
10 model[j].add(Conv2D(64,kernel_size=3,activation='relu'))
11 model[j].add(BatchNormalization())
12 model[j].add(BatchNormalization())
13 model[j].add(BatchNormalization())
14 model[j].add(Conv2D(64,kernel_size=5,strides=2,padding='same',activation='relu'))
15 model[j].add(BatchNormalization())
16 model[j].add(Dropout(0.4))
17 model[j].add(Flatten())
18 model[j].add(Dense(128, activation='relu'))
19 model[j].add(Dropout(0.4))
20 model[j].add(Dense(10, activation='softmax'))
21 model[j].compile(optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"])

In [8]: 1 j=4
2 model[j] = Sequential()
3
4 model[j].add(Conv2D(32,kernel_size=3,activation='relu',input_shape=(28,28,1)))
5 model[j].add(BatchNormalization())
6 model[j].add(Conv2D(32,kernel_size=3,activation='relu'))
7 model[j].add(BatchNormalization())
8 model[j].add(Conv2D(32,kernel_size=5,strides=2,padding='same',activation='relu'))
9 model[j].add(BatchNormalization())
10 model[j].add(Dropout(0.4))
11
12 model[j].add(Conv2D(64,kernel_size=3,activation='relu'))
13 model[j].add(BatchNormalization())
14 model[j].add(Conv2D(64,kernel_size=3,activation='relu'))
15 model[j].add(BatchNormalization())
16 model[j].add(Conv2D(64,kernel_size=5,strides=2,padding='same',activation='relu'))
17 model[j].add(BatchNormalization())
18 model[j].add(Dropout(0.4))
19
20 model[j].add(Flatten())
21 model[j].add(Dense(128, activation='relu'))
22 model[j].add(BatchNormalization())
23 model[j].add(Dropout(0.4))
24 model[j].add(Dense(10, activation='softmax'))
25
26 model[j].compile(optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"])

In [66]: 1 X_train2, X_val2, Y_train2, Y_val2 = train_test_split(X_train, Y_train, test_size = .2)
2 history = [0] * nets
3 names = ["basic","3C23-3C23","32CC52","both+BN","both+BN+DA"]
4 epochs = 35
5 for j in range(nets):
6     history[j] = model[j].fit(X_train2,Y_train2, batch_size=64, epochs = epochs,
7                               validation_data = (X_val2,Y_val2), callbacks=[annealer], verbose=0)
8     print("CNN (%d): Epochs=%d, Train accuracy=%2.5f, Validation accuracy=%2.5f".format(
9         names[j],epochs,max(history[j].history['acc']),max(history[j].history['val_acc']) ))
10
11 datagen = ImageDataGenerator(
12     rotation_range=10,
13     zoom_range = 0.1,
14     width_shift_range=0.1,
15     height_shift_range=0.1)
16
17 j = nets-1
18 history[j] = model[j].fit_generator(datagen.flow(X_train2,Y_train2, batch_size=64),
19                                     epochs = epochs, steps_per_epoch = X_train2.shape[0]//64,
20                                     validation_data = (X_val2,Y_val2), callbacks=[annealer], verbose=0)
21 print("CNN (%d): Epochs=%d, Train accuracy=%2.5f, Validation accuracy=%2.5f".format(
22     names[j],epochs,max(history[j].history['acc']),max(history[j].history['val_acc']) ))
23
24 CNN basic: Epochs=35, Train accuracy=0.99530, Validation accuracy=0.99298
CNN 3C23-3C23: Epochs=35, Train accuracy=0.99687, Validation accuracy=0.99405
CNN 32CC52: Epochs=35, Train accuracy=0.99893, Validation accuracy=0.99345
CNN both+BN: Epochs=35, Train accuracy=0.99917, Validation accuracy=0.99512
CNN both+BN+DA: Epochs=35, Train accuracy=0.99476, Validation accuracy=0.99548

In [67]: 1 plt.figure(figsize=(24,8))
2 for i in range(nets):
3     plt.plot(history[i].history['val_acc'],linestyle=styles[i],linewidth=3)
4     plt.title("Model Accuracy",fontsize=20)
5     plt.ylabel("Accuracy",fontsize=20)
6     plt.xlabel("Epoch",fontsize=20)
7     plt.legend(names, loc='best',fontsize=20)
8     axes=plt.gca()
9     plt.xticks(fontsize=20)
10    plt.yticks(fontsize=20)
11    axes.set_xlim(0.98,1)
```

```
12 plt.savefig("MNIST_CNN (AF).jpg", dpi=300)
13 plt.show()
```

