03_image_classification_with_alexnet

September 29, 2021

1 CIFAR10 Image Classification

Fast-forward to 2012, and we move on to the deeper and more modern AlexNet architecture. We will use the CIFAR10 dataset that uses 60,000 ImageNet samples, compressed to 32x32 pixel resolution (from the original 224x224), but still with three color channels. There are only 10 of the original 1,000 classes.

1.1 Imports

[1]:
```
%matplotlib inline

from pathlib import Path

import numpy as np
import pandas as pd

import tensorflow as tf
from tensorflow.keras.datasets import cifar10
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D,␣
↪MaxPooling2D
from tensorflow.keras.callbacks import ModelCheckpoint, TensorBoard,␣
↪EarlyStopping
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras import backend as K

import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
gpu_devices = tf.config.experimental.list_physical_devices('GPU')
if gpu_devices:
    print('Using GPU')
    tf.config.experimental.set_memory_growth(gpu_devices[0], True)
else:
    print('Using CPU')
```

```
Using CPU
```

[3]:
```
sns.set_style('whitegrid')
np.random.seed(42)
```

[4]:
```
results_path = Path('results', 'cifar10')
if not results_path.exists():
    results_path.mkdir()
```

1.2 Load CIFAR-10 Data

CIFAR10 can also be downloaded from keras, and we similarly rescale the pixel values and one-hot encode the ten class labels.

[5]:
```
# load the pre-shuffled train and test data
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
```

1.2.1 Visualize the First 30 Training Images

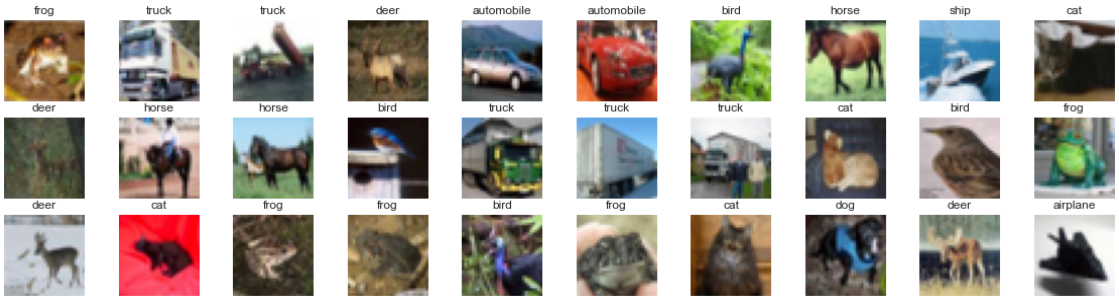
[6]:
```
cifar10_labels = {
    0: 'airplane',
    1: 'automobile',
    2: 'bird',
    3: 'cat',
    4: 'deer',
    5: 'dog',
    6: 'frog',
    7: 'horse',
    8: 'ship',
    9: 'truck'
}
```

[7]:
```
num_classes = len(cifar10_labels)
```

[8]:
```
height, width, channels = X_train.shape[1:]
input_shape = height, width, channels
input_shape
```

[8]:
```
(32, 32, 3)
```

[9]:
```
fig, axes = plt.subplots(nrows=3, ncols=10, figsize=(20, 5))
axes = axes.flatten()
for i, ax in enumerate(axes):
    ax.imshow(np.squeeze(X_train[i]))
    ax.axis('off')
    ax.set_title(cifar10_labels[y_train[i, 0]])
```



1.2.2 Rescale the Images

[10]:
```
# rescale [0,255] --> [0,1]
X_train = X_train.astype('float32') / 255
X_test = X_test.astype('float32') / 255
```

1.2.3 Train-Test split

[11]:
```
X_train, X_valid = X_train[5000:], X_train[:5000]
y_train, y_valid = y_train[5000:], y_train[:5000]
```

[12]:
```
# shape of training set
X_train.shape
```

[12]:
```
(45000, 32, 32, 3)
```

[13]:
```
print(X_train.shape[0], 'train samples')
print(X_test.shape[0], 'test samples')
print(X_valid.shape[0], 'validation samples')
```

```
45000 train samples
10000 test samples
5000 validation samples
```

1.3 Feedforward Neural Network

We first train a two-layer feedforward network on 50,000 training samples for training for 20 epochs to achieve a test accuracy of 44.22%. We also experiment with a three-layer convolutional net with 500K parameters for 67.07% test accuracy.

1.3.1 Model Architecture

[14]:
```
mlp = Sequential([
    Flatten(input_shape=input_shape, name='input'),
    Dense(1000, activation='relu', name='hidden_layer_1'),
    Dropout(0.2, name='droput_1'),
```

```
    Dense(512, activation='relu', name='hidden_layer_2'),
    Dropout(0.2, name='dropout_2'),
    Dense(num_classes, activation='softmax', name='output')
])
```

[15]:
```
mlp.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input (Flatten)              (None, 3072)              0
_________________________________________________________________
hidden_layer_1 (Dense)       (None, 1000)              3073000
_________________________________________________________________
droput_1 (Dropout)           (None, 1000)              0
_________________________________________________________________
hidden_layer_2 (Dense)       (None, 512)               512512
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
output (Dense)               (None, 10)                5130
=================================================================
Total params: 3,590,642
Trainable params: 3,590,642
Non-trainable params: 0
_________________________________________________________________
```

1.3.2 Compile the Model

[16]:
```
mlp.compile(loss='sparse_categorical_crossentropy',
            optimizer='adam',
            metrics=['accuracy'])
```

1.3.3 Define Callbacks

[17]:
```
mlp_path = (results_path / 'mlp.weights.best.hdf5').as_posix()
```

[18]:
```
checkpointer = ModelCheckpoint(filepath=mlp_path,
                               verbose=1,
                               monitor='val_accuracy',
                               save_best_only=True)
```

[19]:
```
tensorboard = TensorBoard(log_dir=results_path / 'logs' / 'mlp',
                          histogram_freq=1,
                          write_graph=True,
                          write_grads=False,
```

```
                                update_freq='epoch')
```

[20]:
```
early_stopping = EarlyStopping(monitor='val_accuracy', patience=10)
```

1.3.4 Train the Model

[21]:
```
batch_size = 32
epochs = 100
```

[22]:
```
mlp_history = mlp.fit(X_train,
                      y_train,
                      batch_size=batch_size,
                      epochs=epochs,
                      validation_data=(X_valid, y_valid),
                      callbacks=[checkpointer, tensorboard, early_stopping],
                      verbose=1,
                      shuffle=True)
```

```
Epoch 1/100
1402/1407 [============================>.] - ETA: 0s - loss: 1.9862 - accuracy:
0.2738
Epoch 00001: val_accuracy improved from -inf to 0.34060, saving model to
results/cifar10/mlp.weights.best.hdf5
1407/1407 [==============================] - 6s 4ms/step - loss: 1.9860 -
accuracy: 0.2737 - val_loss: 1.8299 - val_accuracy: 0.3406
Epoch 2/100
1396/1407 [============================>.] - ETA: 0s - loss: 1.8507 - accuracy:
0.3252
Epoch 00002: val_accuracy improved from 0.34060 to 0.36960, saving model to
results/cifar10/mlp.weights.best.hdf5
1407/1407 [==============================] - 6s 4ms/step - loss: 1.8507 -
accuracy: 0.3253 - val_loss: 1.7528 - val_accuracy: 0.3696
Epoch 3/100
1396/1407 [============================>.] - ETA: 0s - loss: 1.8112 - accuracy:
0.3431
Epoch 00003: val_accuracy improved from 0.36960 to 0.37920, saving model to
results/cifar10/mlp.weights.best.hdf5
1407/1407 [==============================] - 6s 4ms/step - loss: 1.8110 -
accuracy: 0.3433 - val_loss: 1.7304 - val_accuracy: 0.3792
Epoch 4/100
1393/1407 [============================>.] - ETA: 0s - loss: 1.7753 - accuracy:
0.3546
Epoch 00004: val_accuracy improved from 0.37920 to 0.38040, saving model to
results/cifar10/mlp.weights.best.hdf5
1407/1407 [==============================] - 6s 4ms/step - loss: 1.7751 -
accuracy: 0.3547 - val_loss: 1.7400 - val_accuracy: 0.3804
Epoch 5/100
```

```
1405/1407 [============================>.] - ETA: 0s - loss: 1.7524 - accuracy:
0.3606
Epoch 00005: val_accuracy did not improve from 0.38040
1407/1407 [==============================] - 6s 4ms/step - loss: 1.7524 -
accuracy: 0.3606 - val_loss: 1.7227 - val_accuracy: 0.3644
Epoch 6/100
1397/1407 [============================>.] - ETA: 0s - loss: 1.7355 - accuracy:
0.3697
Epoch 00006: val_accuracy improved from 0.38040 to 0.41040, saving model to
results/cifar10/mlp.weights.best.hdf5
1407/1407 [==============================] - 5s 3ms/step - loss: 1.7352 -
accuracy: 0.3701 - val_loss: 1.6512 - val_accuracy: 0.4104
Epoch 7/100
1387/1407 [============================>.] - ETA: 0s - loss: 1.7258 - accuracy:
0.3722
Epoch 00007: val_accuracy did not improve from 0.41040
1407/1407 [==============================] - 4s 3ms/step - loss: 1.7260 -
accuracy: 0.3720 - val_loss: 1.6519 - val_accuracy: 0.4040
Epoch 8/100
1405/1407 [============================>.] - ETA: 0s - loss: 1.7174 - accuracy:
0.3764
Epoch 00008: val_accuracy improved from 0.41040 to 0.41380, saving model to
results/cifar10/mlp.weights.best.hdf5
1407/1407 [==============================] - 4s 3ms/step - loss: 1.7176 -
accuracy: 0.3763 - val_loss: 1.6444 - val_accuracy: 0.4138
Epoch 9/100
1407/1407 [==============================] - ETA: 0s - loss: 1.7079 - accuracy:
0.3780
Epoch 00009: val_accuracy improved from 0.41380 to 0.41440, saving model to
results/cifar10/mlp.weights.best.hdf5
1407/1407 [==============================] - 4s 3ms/step - loss: 1.7079 -
accuracy: 0.3780 - val_loss: 1.6564 - val_accuracy: 0.4144
Epoch 10/100
1405/1407 [============================>.] - ETA: 0s - loss: 1.6986 - accuracy:
0.3818
Epoch 00010: val_accuracy did not improve from 0.41440
1407/1407 [==============================] - 4s 3ms/step - loss: 1.6985 -
accuracy: 0.3818 - val_loss: 1.6394 - val_accuracy: 0.4082
Epoch 11/100
1401/1407 [============================>.] - ETA: 0s - loss: 1.6816 - accuracy:
0.3913
Epoch 00011: val_accuracy did not improve from 0.41440
1407/1407 [==============================] - 4s 3ms/step - loss: 1.6813 -
accuracy: 0.3913 - val_loss: 1.6265 - val_accuracy: 0.4110
Epoch 12/100
1386/1407 [============================>.] - ETA: 0s - loss: 1.6756 - accuracy:
0.3918
Epoch 00012: val_accuracy improved from 0.41440 to 0.41580, saving model to
```

```
results/cifar10/mlp.weights.best.hdf5
1407/1407 [==============================] - 4s 3ms/step - loss: 1.6753 -
accuracy: 0.3918 - val_loss: 1.6300 - val_accuracy: 0.4158
Epoch 13/100
1398/1407 [============================>.] - ETA: 0s - loss: 1.6718 - accuracy:
0.3921
Epoch 00013: val_accuracy did not improve from 0.41580
1407/1407 [==============================] - 4s 3ms/step - loss: 1.6720 -
accuracy: 0.3921 - val_loss: 1.6536 - val_accuracy: 0.4060
Epoch 14/100
1403/1407 [============================>.] - ETA: 0s - loss: 1.6704 - accuracy:
0.3932
Epoch 00014: val_accuracy improved from 0.41580 to 0.43520, saving model to
results/cifar10/mlp.weights.best.hdf5
1407/1407 [==============================] - 4s 3ms/step - loss: 1.6703 -
accuracy: 0.3932 - val_loss: 1.6076 - val_accuracy: 0.4352
Epoch 15/100
1405/1407 [============================>.] - ETA: 0s - loss: 1.6606 - accuracy:
0.3987
Epoch 00015: val_accuracy did not improve from 0.43520
1407/1407 [==============================] - 4s 3ms/step - loss: 1.6605 -
accuracy: 0.3987 - val_loss: 1.6234 - val_accuracy: 0.4220
Epoch 16/100
1389/1407 [============================>.] - ETA: 0s - loss: 1.6513 - accuracy:
0.4005
Epoch 00016: val_accuracy did not improve from 0.43520
1407/1407 [==============================] - 4s 3ms/step - loss: 1.6518 -
accuracy: 0.4003 - val_loss: 1.5963 - val_accuracy: 0.4292
Epoch 17/100
1385/1407 [============================>.] - ETA: 0s - loss: 1.6436 - accuracy:
0.4069
Epoch 00017: val_accuracy did not improve from 0.43520
1407/1407 [==============================] - 4s 3ms/step - loss: 1.6441 -
accuracy: 0.4071 - val_loss: 1.6044 - val_accuracy: 0.4342
Epoch 18/100
1405/1407 [============================>.] - ETA: 0s - loss: 1.6413 - accuracy:
0.4060
Epoch 00018: val_accuracy improved from 0.43520 to 0.44120, saving model to
results/cifar10/mlp.weights.best.hdf5
1407/1407 [==============================] - 4s 3ms/step - loss: 1.6415 -
accuracy: 0.4060 - val_loss: 1.5820 - val_accuracy: 0.4412
Epoch 19/100
1404/1407 [============================>.] - ETA: 0s - loss: 1.6341 - accuracy:
0.4087
Epoch 00019: val_accuracy did not improve from 0.44120
1407/1407 [==============================] - 4s 3ms/step - loss: 1.6342 -
accuracy: 0.4085 - val_loss: 1.6050 - val_accuracy: 0.4214
Epoch 20/100
```

```
1389/1407 [============================>.] - ETA: 0s - loss: 1.6365 - accuracy:
0.4084
Epoch 00020: val_accuracy did not improve from 0.44120
1407/1407 [==============================] - 4s 3ms/step - loss: 1.6372 -
accuracy: 0.4078 - val_loss: 1.5929 - val_accuracy: 0.4292
Epoch 21/100
1404/1407 [============================>.] - ETA: 0s - loss: 1.6283 - accuracy:
0.4106
Epoch 00021: val_accuracy did not improve from 0.44120
1407/1407 [==============================] - 4s 3ms/step - loss: 1.6278 -
accuracy: 0.4109 - val_loss: 1.5937 - val_accuracy: 0.4332
Epoch 22/100
1387/1407 [============================>.] - ETA: 0s - loss: 1.6279 - accuracy:
0.4125
Epoch 00022: val_accuracy improved from 0.44120 to 0.44440, saving model to
results/cifar10/mlp.weights.best.hdf5
1407/1407 [==============================] - 4s 3ms/step - loss: 1.6280 -
accuracy: 0.4121 - val_loss: 1.5746 - val_accuracy: 0.4444
Epoch 23/100
1388/1407 [============================>.] - ETA: 0s - loss: 1.6210 - accuracy:
0.4130
Epoch 00023: val_accuracy did not improve from 0.44440
1407/1407 [==============================] - 4s 3ms/step - loss: 1.6215 -
accuracy: 0.4129 - val_loss: 1.5896 - val_accuracy: 0.4392
Epoch 24/100
1394/1407 [============================>.] - ETA: 0s - loss: 1.6201 - accuracy:
0.4141
Epoch 00024: val_accuracy did not improve from 0.44440
1407/1407 [==============================] - 4s 3ms/step - loss: 1.6195 -
accuracy: 0.4143 - val_loss: 1.5868 - val_accuracy: 0.4326
Epoch 25/100
1389/1407 [============================>.] - ETA: 0s - loss: 1.6142 - accuracy:
0.4199
Epoch 00025: val_accuracy did not improve from 0.44440
1407/1407 [==============================] - 4s 3ms/step - loss: 1.6145 -
accuracy: 0.4196 - val_loss: 1.5785 - val_accuracy: 0.4306
Epoch 26/100
1404/1407 [============================>.] - ETA: 0s - loss: 1.6082 - accuracy:
0.4200
Epoch 00026: val_accuracy did not improve from 0.44440
1407/1407 [==============================] - 4s 3ms/step - loss: 1.6082 -
accuracy: 0.4200 - val_loss: 1.5691 - val_accuracy: 0.4358
Epoch 27/100
1398/1407 [============================>.] - ETA: 0s - loss: 1.6123 - accuracy:
0.4180
Epoch 00027: val_accuracy did not improve from 0.44440
1407/1407 [==============================] - 4s 3ms/step - loss: 1.6124 -
accuracy: 0.4178 - val_loss: 1.5746 - val_accuracy: 0.4370
```

```
Epoch 28/100
1394/1407 [============================>.] - ETA: 0s - loss: 1.6038 - accuracy:
0.4184
Epoch 00028: val_accuracy did not improve from 0.44440
1407/1407 [==============================] - 4s 3ms/step - loss: 1.6039 -
accuracy: 0.4182 - val_loss: 1.5717 - val_accuracy: 0.4438
Epoch 29/100
1401/1407 [============================>.] - ETA: 0s - loss: 1.6009 - accuracy:
0.4227
Epoch 00029: val_accuracy did not improve from 0.44440
1407/1407 [==============================] - 4s 3ms/step - loss: 1.6012 -
accuracy: 0.4228 - val_loss: 1.5908 - val_accuracy: 0.4380
Epoch 30/100
1398/1407 [============================>.] - ETA: 0s - loss: 1.5966 - accuracy:
0.4207
Epoch 00030: val_accuracy did not improve from 0.44440
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5970 -
accuracy: 0.4205 - val_loss: 1.5800 - val_accuracy: 0.4360
Epoch 31/100
1388/1407 [============================>.] - ETA: 0s - loss: 1.6026 - accuracy:
0.4239
Epoch 00031: val_accuracy did not improve from 0.44440
1407/1407 [==============================] - 4s 3ms/step - loss: 1.6027 -
accuracy: 0.4236 - val_loss: 1.5734 - val_accuracy: 0.4370
Epoch 32/100
1398/1407 [============================>.] - ETA: 0s - loss: 1.5999 - accuracy:
0.4240
Epoch 00032: val_accuracy improved from 0.44440 to 0.44840, saving model to
results/cifar10/mlp.weights.best.hdf5
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5998 -
accuracy: 0.4241 - val_loss: 1.5570 - val_accuracy: 0.4484
Epoch 33/100
1398/1407 [============================>.] - ETA: 0s - loss: 1.5912 - accuracy:
0.4258
Epoch 00033: val_accuracy did not improve from 0.44840
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5917 -
accuracy: 0.4258 - val_loss: 1.5770 - val_accuracy: 0.4242
Epoch 34/100
1395/1407 [============================>.] - ETA: 0s - loss: 1.5965 - accuracy:
0.4250
Epoch 00034: val_accuracy did not improve from 0.44840
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5964 -
accuracy: 0.4250 - val_loss: 1.5852 - val_accuracy: 0.4342
Epoch 35/100
1390/1407 [============================>.] - ETA: 0s - loss: 1.5854 - accuracy:
0.4275
Epoch 00035: val_accuracy did not improve from 0.44840
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5858 -
```

```
accuracy: 0.4272 - val_loss: 1.5676 - val_accuracy: 0.4338
Epoch 36/100
1402/1407 [============================>.] - ETA: 0s - loss: 1.5841 - accuracy:
0.4289
Epoch 00036: val_accuracy did not improve from 0.44840
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5836 -
accuracy: 0.4290 - val_loss: 1.5774 - val_accuracy: 0.4316
Epoch 37/100
1390/1407 [============================>.] - ETA: 0s - loss: 1.5835 - accuracy:
0.4286
Epoch 00037: val_accuracy did not improve from 0.44840
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5839 -
accuracy: 0.4283 - val_loss: 1.5791 - val_accuracy: 0.4340
Epoch 38/100
1395/1407 [============================>.] - ETA: 0s - loss: 1.5823 - accuracy:
0.4286
Epoch 00038: val_accuracy did not improve from 0.44840
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5820 -
accuracy: 0.4288 - val_loss: 1.5574 - val_accuracy: 0.4478
Epoch 39/100
1388/1407 [============================>.] - ETA: 0s - loss: 1.5856 - accuracy:
0.4275
Epoch 00039: val_accuracy did not improve from 0.44840
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5859 -
accuracy: 0.4273 - val_loss: 1.5651 - val_accuracy: 0.4330
Epoch 40/100
1389/1407 [============================>.] - ETA: 0s - loss: 1.5787 - accuracy:
0.4314
Epoch 00040: val_accuracy did not improve from 0.44840
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5781 -
accuracy: 0.4314 - val_loss: 1.5556 - val_accuracy: 0.4422
Epoch 41/100
1397/1407 [============================>.] - ETA: 0s - loss: 1.5787 - accuracy:
0.4322
Epoch 00041: val_accuracy improved from 0.44840 to 0.45520, saving model to
results/cifar10/mlp.weights.best.hdf5
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5789 -
accuracy: 0.4318 - val_loss: 1.5452 - val_accuracy: 0.4552
Epoch 42/100
1389/1407 [============================>.] - ETA: 0s - loss: 1.5756 - accuracy:
0.4339
Epoch 00042: val_accuracy did not improve from 0.45520
1407/1407 [==============================] - 3s 2ms/step - loss: 1.5753 -
accuracy: 0.4339 - val_loss: 1.5623 - val_accuracy: 0.4460
Epoch 43/100
1396/1407 [============================>.] - ETA: 0s - loss: 1.5784 - accuracy:
0.4315
Epoch 00043: val_accuracy did not improve from 0.45520
```

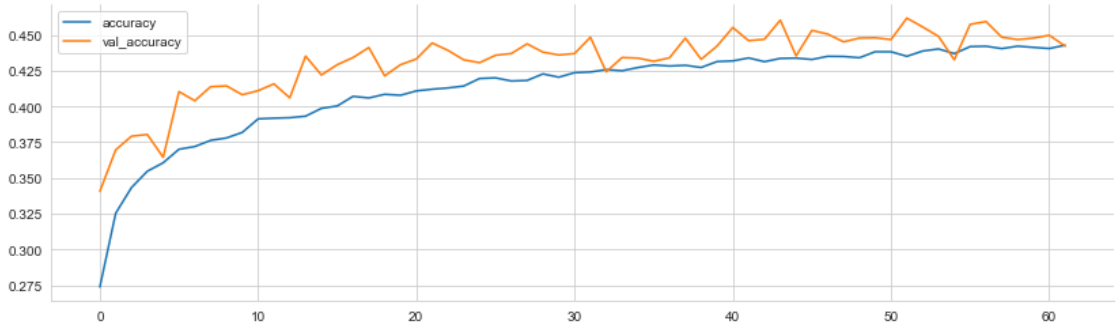
```
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5791 -
accuracy: 0.4313 - val_loss: 1.5775 - val_accuracy: 0.4470
Epoch 44/100
1393/1407 [============================>.] - ETA: 0s - loss: 1.5714 - accuracy:
0.4336
Epoch 00044: val_accuracy improved from 0.45520 to 0.46040, saving model to
results/cifar10/mlp.weights.best.hdf5
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5709 -
accuracy: 0.4336 - val_loss: 1.5385 - val_accuracy: 0.4604
Epoch 45/100
1399/1407 [============================>.] - ETA: 0s - loss: 1.5719 - accuracy:
0.4338
Epoch 00045: val_accuracy did not improve from 0.46040
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5721 -
accuracy: 0.4338 - val_loss: 1.5764 - val_accuracy: 0.4354
Epoch 46/100
1386/1407 [============================>.] - ETA: 0s - loss: 1.5727 - accuracy:
0.4328
Epoch 00046: val_accuracy did not improve from 0.46040
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5725 -
accuracy: 0.4329 - val_loss: 1.5394 - val_accuracy: 0.4532
Epoch 47/100
1402/1407 [============================>.] - ETA: 0s - loss: 1.5674 - accuracy:
0.4347
Epoch 00047: val_accuracy did not improve from 0.46040
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5672 -
accuracy: 0.4351 - val_loss: 1.5785 - val_accuracy: 0.4506
Epoch 48/100
1405/1407 [============================>.] - ETA: 0s - loss: 1.5659 - accuracy:
0.4350
Epoch 00048: val_accuracy did not improve from 0.46040
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5659 -
accuracy: 0.4350 - val_loss: 1.5554 - val_accuracy: 0.4452
Epoch 49/100
1396/1407 [============================>.] - ETA: 0s - loss: 1.5663 - accuracy:
0.4341
Epoch 00049: val_accuracy did not improve from 0.46040
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5663 -
accuracy: 0.4341 - val_loss: 1.5583 - val_accuracy: 0.4478
Epoch 50/100
1401/1407 [============================>.] - ETA: 0s - loss: 1.5610 - accuracy:
0.4384
Epoch 00050: val_accuracy did not improve from 0.46040
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5613 -
accuracy: 0.4383 - val_loss: 1.5428 - val_accuracy: 0.4480
Epoch 51/100
1386/1407 [============================>.] - ETA: 0s - loss: 1.5579 - accuracy:
0.4385
```

```
Epoch 00051: val_accuracy did not improve from 0.46040
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5586 -
accuracy: 0.4382 - val_loss: 1.5635 - val_accuracy: 0.4468
Epoch 52/100
1392/1407 [============================>.] - ETA: 0s - loss: 1.5660 - accuracy:
0.4352
Epoch 00052: val_accuracy improved from 0.46040 to 0.46180, saving model to
results/cifar10/mlp.weights.best.hdf5
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5660 -
accuracy: 0.4351 - val_loss: 1.5402 - val_accuracy: 0.4618
Epoch 53/100
1392/1407 [============================>.] - ETA: 0s - loss: 1.5593 - accuracy:
0.4385
Epoch 00053: val_accuracy did not improve from 0.46180
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5599 -
accuracy: 0.4388 - val_loss: 1.5487 - val_accuracy: 0.4556
Epoch 54/100
1389/1407 [============================>.] - ETA: 0s - loss: 1.5583 - accuracy:
0.4401
Epoch 00054: val_accuracy did not improve from 0.46180
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5585 -
accuracy: 0.4402 - val_loss: 1.5673 - val_accuracy: 0.4490
Epoch 55/100
1390/1407 [============================>.] - ETA: 0s - loss: 1.5567 - accuracy:
0.4368
Epoch 00055: val_accuracy did not improve from 0.46180
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5565 -
accuracy: 0.4370 - val_loss: 1.5689 - val_accuracy: 0.4326
Epoch 56/100
1398/1407 [============================>.] - ETA: 0s - loss: 1.5556 - accuracy:
0.4418
Epoch 00056: val_accuracy did not improve from 0.46180
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5553 -
accuracy: 0.4419 - val_loss: 1.5362 - val_accuracy: 0.4574
Epoch 57/100
1400/1407 [============================>.] - ETA: 0s - loss: 1.5509 - accuracy:
0.4421
Epoch 00057: val_accuracy did not improve from 0.46180
1407/1407 [==============================] - 5s 3ms/step - loss: 1.5507 -
accuracy: 0.4421 - val_loss: 1.5463 - val_accuracy: 0.4594
Epoch 58/100
1397/1407 [============================>.] - ETA: 0s - loss: 1.5535 - accuracy:
0.4405
Epoch 00058: val_accuracy did not improve from 0.46180
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5536 -
accuracy: 0.4405 - val_loss: 1.5584 - val_accuracy: 0.4484
Epoch 59/100
1405/1407 [============================>.] - ETA: 0s - loss: 1.5533 - accuracy:
```

```
0.4422
Epoch 00059: val_accuracy did not improve from 0.46180
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5533 -
accuracy: 0.4422 - val_loss: 1.5610 - val_accuracy: 0.4468
Epoch 60/100
1391/1407 [============================>.] - ETA: 0s - loss: 1.5494 - accuracy:
0.4417
Epoch 00060: val_accuracy did not improve from 0.46180
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5496 -
accuracy: 0.4413 - val_loss: 1.5644 - val_accuracy: 0.4478
Epoch 61/100
1390/1407 [============================>.] - ETA: 0s - loss: 1.5515 - accuracy:
0.4408
Epoch 00061: val_accuracy did not improve from 0.46180
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5516 -
accuracy: 0.4405 - val_loss: 1.5575 - val_accuracy: 0.4498
Epoch 62/100
1398/1407 [============================>.] - ETA: 0s - loss: 1.5487 - accuracy:
0.4430
Epoch 00062: val_accuracy did not improve from 0.46180
1407/1407 [==============================] - 4s 3ms/step - loss: 1.5487 -
accuracy: 0.4428 - val_loss: 1.5913 - val_accuracy: 0.4422
```

1.3.5 Plot CV Results

[23]:
```
pd.DataFrame(mlp_history.history)[['accuracy', 'val_accuracy']].
 ↪plot(figsize=(14, 4))
sns.despine()
```



1.3.6 Load best model

[24]:
```
# load the weights that yielded the best validation accuracy
mlp.load_weights(mlp_path)
```

1.3.7 Test Classification Accuracy

[25]:
```
# evaluate and print test accuracy
mlp_accuracy = mlp.evaluate(X_test, y_test, verbose=0)[1]
print('Test accuracy: {:.2%}'.format(mlp_accuracy))
```

```
Test accuracy: 45.24%
```

1.4 Convolutional Neural Network

[26]:
```
# https://stackoverflow.com/questions/35114376/
 ↪error-when-computing-summaries-in-tensorflow/35117760#35117760
K.clear_session()
```

1.4.1 Model Architecture

[27]:
```
cnn = Sequential([
    Conv2D(filters=16,
           kernel_size=2,
           padding='same',
           activation='relu',
           input_shape=input_shape,
           name='CONV1'),
    MaxPooling2D(pool_size=2, name='POOL1'),
    Conv2D(filters=32,
           kernel_size=2,
           padding='same',
           activation='relu',
           name='CONV2'),
    MaxPooling2D(pool_size=2, name='POOL2'),
    Conv2D(filters=64,
           kernel_size=2,
           padding='same',
           activation='relu',
           name='CONV3'),
    MaxPooling2D(pool_size=2, name='POOL3'),
    Dropout(0.3, name='DROP1'),
    Flatten(name='FLAT1'),
    Dense(500, activation='relu', name='FC1'),
    Dropout(0.4, name='DROP2'),
    Dense(10, activation='softmax', name='FC2')
])
```

[28]:
```
cnn.summary()
```

```
Model: "sequential"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
```

```
=================================================================
CONV1 (Conv2D)               (None, 32, 32, 16)        208
_________________________________________________________________
POOL1 (MaxPooling2D)         (None, 16, 16, 16)        0
_________________________________________________________________
CONV2 (Conv2D)               (None, 16, 16, 32)        2080
_________________________________________________________________
POOL2 (MaxPooling2D)         (None, 8, 8, 32)          0
_________________________________________________________________
CONV3 (Conv2D)               (None, 8, 8, 64)          8256
_________________________________________________________________
POOL3 (MaxPooling2D)         (None, 4, 4, 64)          0
_________________________________________________________________
DROP1 (Dropout)              (None, 4, 4, 64)          0
_________________________________________________________________
FLAT1 (Flatten)              (None, 1024)              0
_________________________________________________________________
FC1 (Dense)                  (None, 500)               512500
_________________________________________________________________
DROP2 (Dropout)              (None, 500)               0
_________________________________________________________________
FC2 (Dense)                  (None, 10)                5010
=================================================================
Total params: 528,054
Trainable params: 528,054
Non-trainable params: 0
_________________________________________________________________
```

1.4.2 Compile the Model

[29]:
```
cnn.compile(loss='sparse_categorical_crossentropy',
            optimizer='adam',
            metrics=['accuracy'])
```

1.4.3 Define Callbacks

[30]:
```
cnn_path = (results_path / 'cnn.weights.best.hdf5').as_posix()
```

[31]:
```
checkpointer = ModelCheckpoint(filepath=cnn_path,
                               verbose=1,
                               monitor='val_accuracy',
                               save_best_only=True)
```

[32]:
```
tensorboard = TensorBoard(log_dir=results_path / 'logs' / 'cnn',
                          histogram_freq=1,
                          write_graph=True,
                          write_grads=False,
```

```
                                update_freq='epoch')
```

[33]:
```
early_stopping = EarlyStopping(monitor='val_accuracy', patience=10)
```

1.4.4 Train the Model

[34]:
```
batch_size = 32
epochs = 100
```

[35]:
```
cnn_history = cnn.fit(X_train,
                      y_train,
                      batch_size=batch_size,
                      epochs=epochs,
                      validation_data=(X_valid, y_valid),
                      callbacks=[checkpointer, tensorboard, early_stopping],
                      verbose=2,
                      shuffle=True)
```

```
Epoch 1/100

Epoch 00001: val_accuracy improved from -inf to 0.53480, saving model to
results/cifar10/cnn.weights.best.hdf5
1407/1407 - 5s - loss: 1.5758 - accuracy: 0.4249 - val_loss: 1.2937 -
val_accuracy: 0.5348
Epoch 2/100

Epoch 00002: val_accuracy improved from 0.53480 to 0.60000, saving model to
results/cifar10/cnn.weights.best.hdf5
1407/1407 - 4s - loss: 1.2683 - accuracy: 0.5436 - val_loss: 1.1202 -
val_accuracy: 0.6000
Epoch 3/100

Epoch 00003: val_accuracy improved from 0.60000 to 0.64120, saving model to
results/cifar10/cnn.weights.best.hdf5
1407/1407 - 5s - loss: 1.1385 - accuracy: 0.5918 - val_loss: 1.0183 -
val_accuracy: 0.6412
Epoch 4/100

Epoch 00004: val_accuracy improved from 0.64120 to 0.65340, saving model to
results/cifar10/cnn.weights.best.hdf5
1407/1407 - 5s - loss: 1.0481 - accuracy: 0.6269 - val_loss: 0.9816 -
val_accuracy: 0.6534
Epoch 5/100

Epoch 00005: val_accuracy improved from 0.65340 to 0.68180, saving model to
results/cifar10/cnn.weights.best.hdf5
1407/1407 - 5s - loss: 0.9824 - accuracy: 0.6506 - val_loss: 0.9043 -
```

```
val_accuracy: 0.6818
Epoch 6/100

Epoch 00006: val_accuracy improved from 0.68180 to 0.68680, saving model to
results/cifar10/cnn.weights.best.hdf5
1407/1407 - 4s - loss: 0.9298 - accuracy: 0.6710 - val_loss: 0.8882 -
val_accuracy: 0.6868
Epoch 7/100

Epoch 00007: val_accuracy improved from 0.68680 to 0.71380, saving model to
results/cifar10/cnn.weights.best.hdf5
1407/1407 - 5s - loss: 0.8844 - accuracy: 0.6866 - val_loss: 0.8158 -
val_accuracy: 0.7138
Epoch 8/100

Epoch 00008: val_accuracy improved from 0.71380 to 0.72500, saving model to
results/cifar10/cnn.weights.best.hdf5
1407/1407 - 5s - loss: 0.8484 - accuracy: 0.6981 - val_loss: 0.8081 -
val_accuracy: 0.7250
Epoch 9/100

Epoch 00009: val_accuracy did not improve from 0.72500
1407/1407 - 5s - loss: 0.8197 - accuracy: 0.7098 - val_loss: 0.7862 -
val_accuracy: 0.7228
Epoch 10/100

Epoch 00010: val_accuracy improved from 0.72500 to 0.73920, saving model to
results/cifar10/cnn.weights.best.hdf5
1407/1407 - 5s - loss: 0.7854 - accuracy: 0.7228 - val_loss: 0.7704 -
val_accuracy: 0.7392
Epoch 11/100

Epoch 00011: val_accuracy improved from 0.73920 to 0.74360, saving model to
results/cifar10/cnn.weights.best.hdf5
1407/1407 - 5s - loss: 0.7566 - accuracy: 0.7334 - val_loss: 0.7560 -
val_accuracy: 0.7436
Epoch 12/100

Epoch 00012: val_accuracy did not improve from 0.74360
1407/1407 - 5s - loss: 0.7425 - accuracy: 0.7370 - val_loss: 0.7834 -
val_accuracy: 0.7228
Epoch 13/100

Epoch 00013: val_accuracy did not improve from 0.74360
1407/1407 - 5s - loss: 0.7146 - accuracy: 0.7458 - val_loss: 0.7438 -
val_accuracy: 0.7404
Epoch 14/100
```

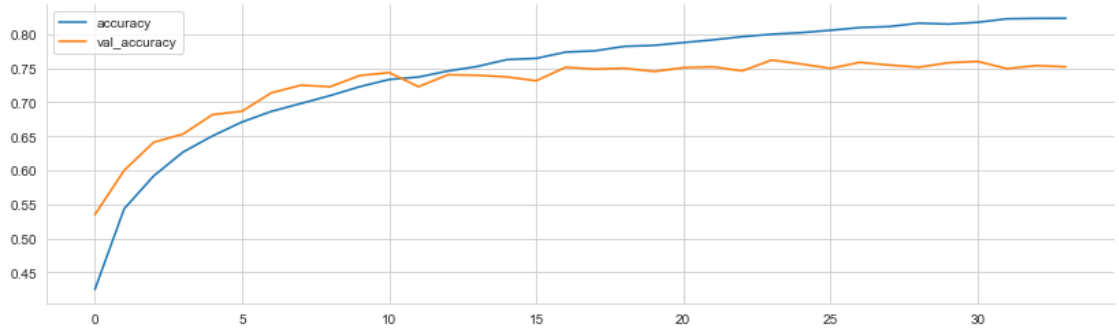
```
Epoch 00014: val_accuracy did not improve from 0.74360
1407/1407 - 4s - loss: 0.6945 - accuracy: 0.7526 - val_loss: 0.7510 -
val_accuracy: 0.7396
Epoch 15/100

Epoch 00015: val_accuracy did not improve from 0.74360
1407/1407 - 5s - loss: 0.6722 - accuracy: 0.7626 - val_loss: 0.7723 -
val_accuracy: 0.7372
Epoch 16/100

Epoch 00016: val_accuracy did not improve from 0.74360
1407/1407 - 4s - loss: 0.6586 - accuracy: 0.7644 - val_loss: 0.7758 -
val_accuracy: 0.7316
Epoch 17/100

Epoch 00017: val_accuracy improved from 0.74360 to 0.75140, saving model to
results/cifar10/cnn.weights.best.hdf5
1407/1407 - 5s - loss: 0.6427 - accuracy: 0.7736 - val_loss: 0.7227 -
val_accuracy: 0.7514
Epoch 18/100

Epoch 00018: val_accuracy did not improve from 0.75140
1407/1407 - 4s - loss: 0.6261 - accuracy: 0.7756 - val_loss: 0.7314 -
val_accuracy: 0.7488
Epoch 19/100

Epoch 00019: val_accuracy did not improve from 0.75140
1407/1407 - 5s - loss: 0.6130 - accuracy: 0.7821 - val_loss: 0.7269 -
val_accuracy: 0.7500
Epoch 20/100

Epoch 00020: val_accuracy did not improve from 0.75140
1407/1407 - 4s - loss: 0.6058 - accuracy: 0.7835 - val_loss: 0.7349 -
val_accuracy: 0.7452
Epoch 21/100

Epoch 00021: val_accuracy did not improve from 0.75140
1407/1407 - 5s - loss: 0.5940 - accuracy: 0.7876 - val_loss: 0.7228 -
val_accuracy: 0.7508
Epoch 22/100

Epoch 00022: val_accuracy improved from 0.75140 to 0.75200, saving model to
results/cifar10/cnn.weights.best.hdf5
1407/1407 - 5s - loss: 0.5836 - accuracy: 0.7916 - val_loss: 0.7152 -
val_accuracy: 0.7520
Epoch 23/100

Epoch 00023: val_accuracy did not improve from 0.75200
```

```
1407/1407 - 5s - loss: 0.5667 - accuracy: 0.7963 - val_loss: 0.7337 -
val_accuracy: 0.7460
Epoch 24/100

Epoch 00024: val_accuracy improved from 0.75200 to 0.76200, saving model to
results/cifar10/cnn.weights.best.hdf5
1407/1407 - 4s - loss: 0.5579 - accuracy: 0.7999 - val_loss: 0.7081 -
val_accuracy: 0.7620
Epoch 25/100

Epoch 00025: val_accuracy did not improve from 0.76200
1407/1407 - 4s - loss: 0.5502 - accuracy: 0.8022 - val_loss: 0.7219 -
val_accuracy: 0.7562
Epoch 26/100

Epoch 00026: val_accuracy did not improve from 0.76200
1407/1407 - 4s - loss: 0.5397 - accuracy: 0.8057 - val_loss: 0.7275 -
val_accuracy: 0.7498
Epoch 27/100

Epoch 00027: val_accuracy did not improve from 0.76200
1407/1407 - 5s - loss: 0.5332 - accuracy: 0.8097 - val_loss: 0.7031 -
val_accuracy: 0.7586
Epoch 28/100

Epoch 00028: val_accuracy did not improve from 0.76200
1407/1407 - 4s - loss: 0.5243 - accuracy: 0.8112 - val_loss: 0.7269 -
val_accuracy: 0.7546
Epoch 29/100

Epoch 00029: val_accuracy did not improve from 0.76200
1407/1407 - 4s - loss: 0.5174 - accuracy: 0.8162 - val_loss: 0.7363 -
val_accuracy: 0.7514
Epoch 30/100

Epoch 00030: val_accuracy did not improve from 0.76200
1407/1407 - 4s - loss: 0.5137 - accuracy: 0.8148 - val_loss: 0.7098 -
val_accuracy: 0.7580
Epoch 31/100

Epoch 00031: val_accuracy did not improve from 0.76200
1407/1407 - 5s - loss: 0.5054 - accuracy: 0.8174 - val_loss: 0.7157 -
val_accuracy: 0.7600
Epoch 32/100

Epoch 00032: val_accuracy did not improve from 0.76200
1407/1407 - 5s - loss: 0.4973 - accuracy: 0.8225 - val_loss: 0.7246 -
val_accuracy: 0.7494
```

```
Epoch 33/100

Epoch 00033: val_accuracy did not improve from 0.76200
1407/1407 - 4s - loss: 0.4918 - accuracy: 0.8232 - val_loss: 0.7304 -
val_accuracy: 0.7538
Epoch 34/100

Epoch 00034: val_accuracy did not improve from 0.76200
1407/1407 - 5s - loss: 0.4905 - accuracy: 0.8233 - val_loss: 0.7392 -
val_accuracy: 0.7520
```

1.4.5 Plot CV Results

[36]:
```
pd.DataFrame(cnn_history.history)[['accuracy',
                                   'val_accuracy']].plot(figsize=(14, 4))
sns.despine();
```



1.4.6 Load best model

[37]:
```
cnn.load_weights(cnn_path)
```

1.4.7 Test set accuracy

[38]:
```
cnn_accuracy = cnn.evaluate(X_test, y_test, verbose=0)[1]
print('Accuracy: {:.2%}'.format(cnn_accuracy))
```

```
Accuracy: 75.15%
```

1.4.8 Evaluate Predictions

[39]:
```
y_hat = cnn.predict(X_test)
```

[40]:
```
fig, axes = plt.subplots(nrows=4, ncols=8, figsize=(20, 8))
axes = axes.flatten()
images = np.random.choice(X_test.shape[0], size=32, replace=False)
```

```
for i, (ax, idx) in enumerate(zip(axes, images)):
    ax.imshow(np.squeeze(X_test[idx]))
    ax.axis('off')
    pred_idx, true_idx = np.argmax(y_hat[idx]), np.argmax(y_test[idx])
    if pred_idx == true_idx:
        ax.set_title('{} ( )'.format(cifar10_labels[pred_idx]), color="green")
    else:
        ax.set_title("{} ({})".format(cifar10_labels[pred_idx],
                                      cifar10_labels[true_idx]),
                     color='red')
```

```
/home/stefan/.pyenv/versions/miniconda3-latest/envs/ml4t-dl/lib/python3.7/site-
packages/matplotlib/backends/backend_agg.py:214: RuntimeWarning: Glyph 10003
missing from current font.
  font.set_text(s, 0.0, flags=flags)
/home/stefan/.pyenv/versions/miniconda3-latest/envs/ml4t-dl/lib/python3.7/site-
packages/matplotlib/backends/backend_agg.py:183: RuntimeWarning: Glyph 10003
missing from current font.
  font.set_text(s, 0, flags=flags)
```

1.5 CNN with Image Augmentation

A common trick to enhance performance is to artificially increase the size of the training set by creating synthetic data. This involves randomly shifting or horizontally flipping the image, or introducing noise into the image.

1.5.1 Create and configure augmented image generator

Keras includes an ImageDataGenerator for this purpose that we can configure and fit to the training data as follows:

[41]:
```
datagen = ImageDataGenerator(
    width_shift_range=0.1,  # randomly horizontal shift
    height_shift_range=0.1,  # randomly vertial shift
    horizontal_flip=True)  # randomly horizontalflip
```

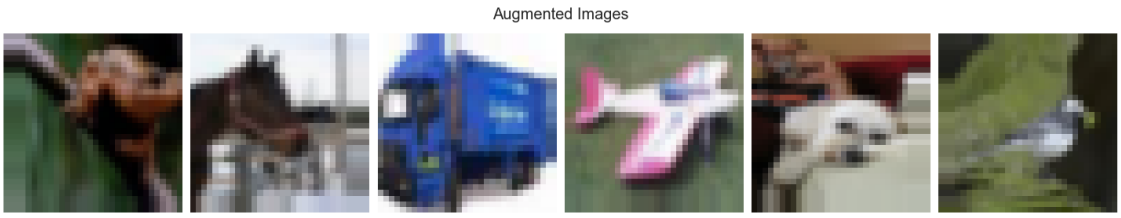
[42]:
```
# fit augmented image generator on data
datagen.fit(X_train)
```

1.5.2 Visualize subset of training data

The result shows how the augmented images have been altered in various ways as expected:

[43]:
```
n_images = 6
x_train_subset = X_train[:n_images]
```

[44]:
```
# original images
fig, axes = plt.subplots(nrows=1, ncols=n_images, figsize=(20, 4))
for i, (ax, img) in enumerate(zip(axes, x_train_subset)):
    ax.imshow(img)
    ax.axis('off')
fig.suptitle('Subset of Original Training Images', fontsize=20)
fig.tight_layout()
fig.subplots_adjust(top=.9)
fig.savefig(results_path / 'original_images')

# augmented images
fig, axes = plt.subplots(nrows=1, ncols=n_images, figsize=(20, 4))
for x_batch in datagen.flow(x_train_subset, batch_size=n_images,
                            shuffle=False):
    for i, ax in enumerate(axes):
        ax.imshow(x_batch[i])
        ax.axis('off')
#     fig.suptitle('Augmented Images', fontsize=20)
    break
fig.suptitle('Augmented Images', fontsize=20)
fig.tight_layout()
fig.subplots_adjust(top=.9)
fig.savefig(results_path / 'augmented_images')
```



1.5.3 Define Callbacks

[45]:
```
K.clear_session()
```

[46]:
```
cnn_aug_path = (results_path / 'augmented.cnn.weights.best.hdf5').as_posix()
```

[47]:
```
checkpointer = ModelCheckpoint(filepath=cnn_aug_path,
                               verbose=1,
                               monitor='val_accuracy',
                               save_best_only=True)
```

[48]:
```
tensorboard = TensorBoard(log_dir=results_path / 'logs' / 'cnn_aug',
                          histogram_freq=1,
                          write_graph=True,
                          write_grads=False,
                          update_freq='epoch')
```

[49]:
```
early_stopping = EarlyStopping(monitor='val_accuracy',
                               patience=10)
```

1.5.4 Train Augmented Images

[50]:
```
batch_size = 32
epochs = 100
```

[51]:
```
cnn_aug_history = cnn.fit(datagen.flow(X_train, y_train, batch_size=batch_size),
                          steps_per_epoch=X_train.shape[0] // batch_size,
                          epochs=epochs,
                          validation_data=(X_valid, y_valid),
                          callbacks=[checkpointer, tensorboard, early_stopping],
                          verbose=2)
```

```
Epoch 1/100

Epoch 00001: val_accuracy improved from -inf to 0.73640, saving model to
results/cifar10/augmented.cnn.weights.best.hdf5
1406/1406 - 16s - loss: 0.9741 - accuracy: 0.6619 - val_loss: 0.7585 -
val_accuracy: 0.7364
```

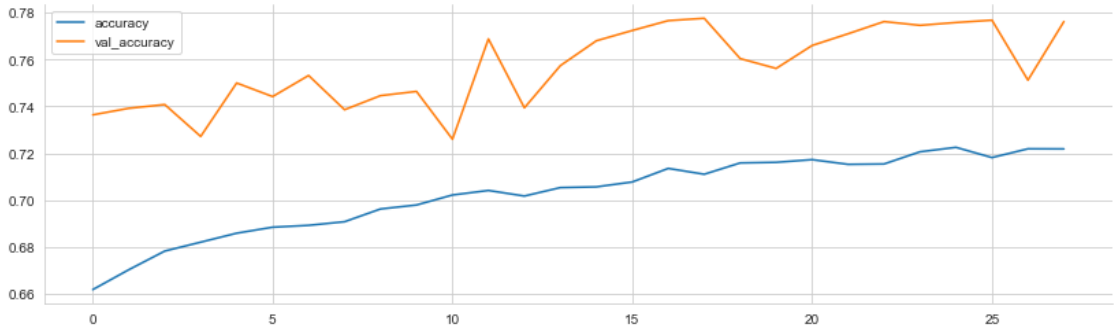
```
Epoch 2/100

Epoch 00002: val_accuracy improved from 0.73640 to 0.73920, saving model to
results/cifar10/augmented.cnn.weights.best.hdf5
1406/1406 - 16s - loss: 0.9427 - accuracy: 0.6703 - val_loss: 0.7555 -
val_accuracy: 0.7392
Epoch 3/100

Epoch 00003: val_accuracy improved from 0.73920 to 0.74080, saving model to
results/cifar10/augmented.cnn.weights.best.hdf5
1406/1406 - 16s - loss: 0.9212 - accuracy: 0.6783 - val_loss: 0.7572 -
val_accuracy: 0.7408
Epoch 4/100

Epoch 00004: val_accuracy did not improve from 0.74080
1406/1406 - 16s - loss: 0.9098 - accuracy: 0.6821 - val_loss: 0.7803 -
val_accuracy: 0.7272
Epoch 5/100

Epoch 00005: val_accuracy improved from 0.74080 to 0.75000, saving model to
results/cifar10/augmented.cnn.weights.best.hdf5
1406/1406 - 16s - loss: 0.9006 - accuracy: 0.6859 - val_loss: 0.7293 -
val_accuracy: 0.7500
Epoch 6/100

Epoch 00006: val_accuracy did not improve from 0.75000
1406/1406 - 16s - loss: 0.8923 - accuracy: 0.6885 - val_loss: 0.7486 -
val_accuracy: 0.7442
Epoch 7/100

Epoch 00007: val_accuracy improved from 0.75000 to 0.75320, saving model to
results/cifar10/augmented.cnn.weights.best.hdf5
1406/1406 - 17s - loss: 0.8807 - accuracy: 0.6893 - val_loss: 0.7524 -
val_accuracy: 0.7532
Epoch 8/100

Epoch 00008: val_accuracy did not improve from 0.75320
1406/1406 - 16s - loss: 0.8761 - accuracy: 0.6908 - val_loss: 0.7420 -
val_accuracy: 0.7386
Epoch 9/100

Epoch 00009: val_accuracy did not improve from 0.75320
1406/1406 - 17s - loss: 0.8691 - accuracy: 0.6963 - val_loss: 0.7364 -
val_accuracy: 0.7446
Epoch 10/100

Epoch 00010: val_accuracy did not improve from 0.75320
1406/1406 - 18s - loss: 0.8596 - accuracy: 0.6979 - val_loss: 0.7214 -
```

```
val_accuracy: 0.7464
Epoch 11/100

Epoch 00011: val_accuracy did not improve from 0.75320
1406/1406 - 18s - loss: 0.8519 - accuracy: 0.7022 - val_loss: 0.8088 -
val_accuracy: 0.7260
Epoch 12/100

Epoch 00012: val_accuracy improved from 0.75320 to 0.76880, saving model to
results/cifar10/augmented.cnn.weights.best.hdf5
1406/1406 - 18s - loss: 0.8434 - accuracy: 0.7041 - val_loss: 0.6818 -
val_accuracy: 0.7688
Epoch 13/100

Epoch 00013: val_accuracy did not improve from 0.76880
1406/1406 - 18s - loss: 0.8420 - accuracy: 0.7018 - val_loss: 0.7699 -
val_accuracy: 0.7394
Epoch 14/100

Epoch 00014: val_accuracy did not improve from 0.76880
1406/1406 - 18s - loss: 0.8419 - accuracy: 0.7054 - val_loss: 0.7041 -
val_accuracy: 0.7574
Epoch 15/100

Epoch 00015: val_accuracy did not improve from 0.76880
1406/1406 - 18s - loss: 0.8388 - accuracy: 0.7057 - val_loss: 0.6848 -
val_accuracy: 0.7680
Epoch 16/100

Epoch 00016: val_accuracy improved from 0.76880 to 0.77240, saving model to
results/cifar10/augmented.cnn.weights.best.hdf5
1406/1406 - 18s - loss: 0.8334 - accuracy: 0.7078 - val_loss: 0.6881 -
val_accuracy: 0.7724
Epoch 17/100

Epoch 00017: val_accuracy improved from 0.77240 to 0.77660, saving model to
results/cifar10/augmented.cnn.weights.best.hdf5
1406/1406 - 18s - loss: 0.8175 - accuracy: 0.7136 - val_loss: 0.6642 -
val_accuracy: 0.7766
Epoch 18/100

Epoch 00018: val_accuracy improved from 0.77660 to 0.77760, saving model to
results/cifar10/augmented.cnn.weights.best.hdf5
1406/1406 - 18s - loss: 0.8271 - accuracy: 0.7111 - val_loss: 0.6709 -
val_accuracy: 0.7776
Epoch 19/100

Epoch 00019: val_accuracy did not improve from 0.77760
```

```
1406/1406 - 18s - loss: 0.8147 - accuracy: 0.7159 - val_loss: 0.7125 -
val_accuracy: 0.7604
Epoch 20/100

Epoch 00020: val_accuracy did not improve from 0.77760
1406/1406 - 18s - loss: 0.8157 - accuracy: 0.7162 - val_loss: 0.7128 -
val_accuracy: 0.7562
Epoch 21/100

Epoch 00021: val_accuracy did not improve from 0.77760
1406/1406 - 17s - loss: 0.8106 - accuracy: 0.7173 - val_loss: 0.6939 -
val_accuracy: 0.7660
Epoch 22/100

Epoch 00022: val_accuracy did not improve from 0.77760
1406/1406 - 18s - loss: 0.8139 - accuracy: 0.7153 - val_loss: 0.6676 -
val_accuracy: 0.7710
Epoch 23/100

Epoch 00023: val_accuracy did not improve from 0.77760
1406/1406 - 18s - loss: 0.8091 - accuracy: 0.7154 - val_loss: 0.6532 -
val_accuracy: 0.7762
Epoch 24/100

Epoch 00024: val_accuracy did not improve from 0.77760
1406/1406 - 18s - loss: 0.7979 - accuracy: 0.7206 - val_loss: 0.6614 -
val_accuracy: 0.7746
Epoch 25/100

Epoch 00025: val_accuracy did not improve from 0.77760
1406/1406 - 18s - loss: 0.7987 - accuracy: 0.7225 - val_loss: 0.6668 -
val_accuracy: 0.7758
Epoch 26/100

Epoch 00026: val_accuracy did not improve from 0.77760
1406/1406 - 17s - loss: 0.8037 - accuracy: 0.7182 - val_loss: 0.6580 -
val_accuracy: 0.7768
Epoch 27/100

Epoch 00027: val_accuracy did not improve from 0.77760
1406/1406 - 17s - loss: 0.7954 - accuracy: 0.7219 - val_loss: 0.7184 -
val_accuracy: 0.7512
Epoch 28/100

Epoch 00028: val_accuracy did not improve from 0.77760
1406/1406 - 18s - loss: 0.7945 - accuracy: 0.7219 - val_loss: 0.6706 -
val_accuracy: 0.7762
```

1.5.5 Plot CV Result

[52]:
```
pd.DataFrame(cnn_aug_history.history)[['accuracy',
                                       'val_accuracy']].plot(figsize=(14, 4))
sns.despine();
```



1.5.6 Load best model

[53]:
```
cnn.load_weights(cnn_aug_path)
```

1.5.7 Test set accuracy

The test accuracy for the three-layer CNN improves markedly to 74.79% after training on the larger, augmented data.

[54]:
```
cnn_aug_accuracy = cnn.evaluate(X_test, y_test, verbose=0)[1]
print('Test Accuracy: {:.2%}'.format(cnn_aug_accuracy))
```

```
Test Accuracy: 76.23%
```

1.6 AlexNet

We also need to simplify the AlexNet architecture in response to the lower dimensionality of CIFAR10 images relative to the ImageNet samples used in the competition. We use the original number of filters but make them smaller (see notebook for implementation). The summary shows the five convolutional layers followed by two fully-connected layers with frequent use of batch normalization, for a total of 21.5 million parameters:

1.6.1 Define Architecture

[55]:
```
K.clear_session()
```

[56]:
```
alexnet = Sequential([

    # 1st Convolutional Layer
    Conv2D(96, (3, 3),
```

```
            strides=(2, 2),
            activation='relu',
            padding='same',
            input_shape=input_shape,
            name='CONV_1'),
    MaxPooling2D(pool_size=(2, 2), strides=(2, 2), name='POOL_1'),
    BatchNormalization(name='NORM_1'),

    # 2nd Convolutional Layer
    Conv2D(filters=256,
            kernel_size=(5, 5),
            padding='same',
            activation='relu',
            name='CONV2'),
    MaxPooling2D(pool_size=(3, 3), strides=(2, 2), name='POOL2'),
    BatchNormalization(name='NORM_2'),

    # 3rd Convolutional Layer
    Conv2D(filters=384,
            kernel_size=(3, 3),
            padding='same',
            activation='relu',
            name='CONV3'),
    # 4th Convolutional Layer
    Conv2D(filters=384,
            kernel_size=(3, 3),
            padding='same',
            activation='relu',
            name='CONV4'),
    # 5th Convolutional Layer
    Conv2D(filters=256,
            kernel_size=(3, 3),
            padding='same',
            activation='relu',
            name='CONV5'),
    MaxPooling2D(pool_size=(3, 3), strides=(2, 2), name='POOL5'),
    BatchNormalization(name='NORM_5'),

    # Fully Connected Layers
    Flatten(name='FLAT'),
    Dense(4096, input_shape=(32 * 32 * 3, ), activation='relu', name='FC1'),
    Dropout(0.4, name='DROP1'),
    Dense(4096, activation='relu', name='FC2'),
    Dropout(0.4, name='DROP2'),
    Dense(num_classes, activation='softmax')
])
```

[57]:
```
alexnet.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
CONV_1 (Conv2D)              (None, 16, 16, 96)        2688
_________________________________________________________________
POOL_1 (MaxPooling2D)        (None, 8, 8, 96)          0
_________________________________________________________________
NORM_1 (BatchNormalization)  (None, 8, 8, 96)          384
_________________________________________________________________
CONV2 (Conv2D)               (None, 8, 8, 256)         614656
_________________________________________________________________
POOL2 (MaxPooling2D)         (None, 3, 3, 256)         0
_________________________________________________________________
NORM_2 (BatchNormalization)  (None, 3, 3, 256)         1024
_________________________________________________________________
CONV3 (Conv2D)               (None, 3, 3, 384)         885120
_________________________________________________________________
CONV4 (Conv2D)               (None, 3, 3, 384)         1327488
_________________________________________________________________
CONV5 (Conv2D)               (None, 3, 3, 256)         884992
_________________________________________________________________
POOL5 (MaxPooling2D)         (None, 1, 1, 256)         0
_________________________________________________________________
NORM_5 (BatchNormalization)  (None, 1, 1, 256)         1024
_________________________________________________________________
FLAT (Flatten)               (None, 256)               0
_________________________________________________________________
FC1 (Dense)                  (None, 4096)              1052672
_________________________________________________________________
DROP1 (Dropout)              (None, 4096)              0
_________________________________________________________________
FC2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
DROP2 (Dropout)              (None, 4096)              0
_________________________________________________________________
dense (Dense)                (None, 10)                40970
=================================================================
Total params: 21,592,330
Trainable params: 21,591,114
Non-trainable params: 1,216
_________________________________________________________________
```

1.6.2 Compile Model

[58]:
```
alexnet.compile(loss='sparse_categorical_crossentropy',
                optimizer='adam',
                metrics=['accuracy'])
```

1.6.3 Define Callbacks

[59]:
```
alexnet_path = (results_path / 'alexnet.weights.best.hdf5').as_posix()
```

[60]:
```
checkpointer = ModelCheckpoint(filepath=alexnet_path,
                               verbose=1,
                               monitor='val_accuracy',
                               save_best_only=True)
```

[61]:
```
tensorboard = TensorBoard(log_dir=results_path / 'logs' / 'alexnet',
                          histogram_freq=1,
                          write_graph=True,
                          write_grads=False,
                          update_freq='epoch')
```

[62]:
```
early_stopping = EarlyStopping(monitor='val_accuracy',
                               mode='max',
                               patience=10)
```

1.6.4 Train Model

[63]:
```
batch_size = 32
epochs = 100
```

[64]:
```
alex_history = alexnet.fit(X_train,
                           y_train,
                           batch_size=batch_size,
                           epochs=epochs,
                           validation_data=(X_valid,
                                            y_valid),
                           callbacks=[checkpointer,
                                      tensorboard,
                                      early_stopping],
                           verbose=1)
```

```
Epoch 1/100
1407/1407 [==============================] - ETA: 0s - loss: 1.6061 - accuracy:
0.4240
Epoch 00001: val_accuracy improved from -inf to 0.55160, saving model to
results/cifar10/alexnet.weights.best.hdf5
1407/1407 [==============================] - 27s 19ms/step - loss: 1.6061 -
```

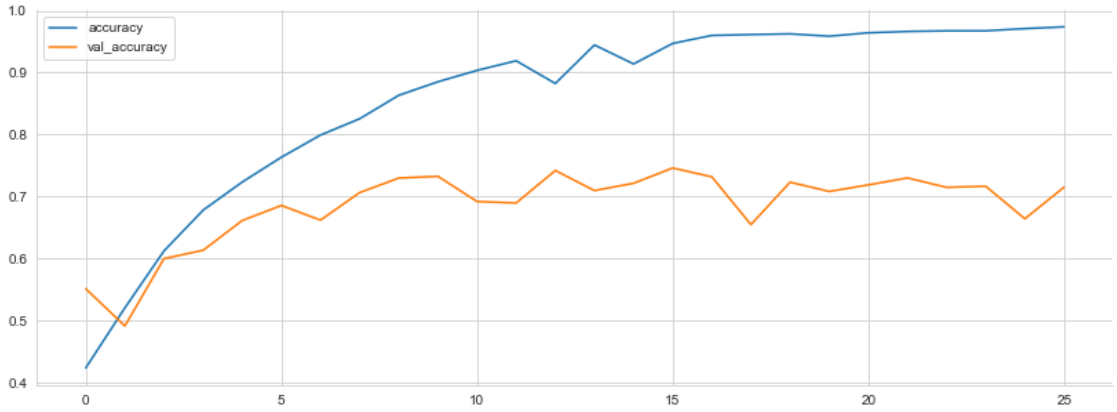
```
accuracy: 0.4240 - val_loss: 1.2629 - val_accuracy: 0.5516
Epoch 2/100
1406/1407 [============================>.] - ETA: 0s - loss: 1.3583 - accuracy:
0.5207
Epoch 00002: val_accuracy did not improve from 0.55160
1407/1407 [==============================] - 26s 18ms/step - loss: 1.3581 -
accuracy: 0.5208 - val_loss: 1.4801 - val_accuracy: 0.4916
Epoch 3/100
1407/1407 [==============================] - ETA: 0s - loss: 1.1069 - accuracy:
0.6126
Epoch 00003: val_accuracy improved from 0.55160 to 0.60020, saving model to
results/cifar10/alexnet.weights.best.hdf5
1407/1407 [==============================] - 35s 25ms/step - loss: 1.1069 -
accuracy: 0.6126 - val_loss: 1.1564 - val_accuracy: 0.6002
Epoch 4/100
1407/1407 [==============================] - ETA: 0s - loss: 0.9455 - accuracy:
0.6782
Epoch 00004: val_accuracy improved from 0.60020 to 0.61360, saving model to
results/cifar10/alexnet.weights.best.hdf5
1407/1407 [==============================] - 34s 24ms/step - loss: 0.9455 -
accuracy: 0.6782 - val_loss: 1.1127 - val_accuracy: 0.6136
Epoch 5/100
1405/1407 [============================>.] - ETA: 0s - loss: 0.8184 - accuracy:
0.7237
Epoch 00005: val_accuracy improved from 0.61360 to 0.66160, saving model to
results/cifar10/alexnet.weights.best.hdf5
1407/1407 [==============================] - 34s 24ms/step - loss: 0.8185 -
accuracy: 0.7236 - val_loss: 0.9998 - val_accuracy: 0.6616
Epoch 6/100
1407/1407 [==============================] - ETA: 0s - loss: 0.7059 - accuracy:
0.7636
Epoch 00006: val_accuracy improved from 0.66160 to 0.68580, saving model to
results/cifar10/alexnet.weights.best.hdf5
1407/1407 [==============================] - 34s 24ms/step - loss: 0.7059 -
accuracy: 0.7636 - val_loss: 0.9658 - val_accuracy: 0.6858
Epoch 7/100
1405/1407 [============================>.] - ETA: 0s - loss: 0.6020 - accuracy:
0.7994
Epoch 00007: val_accuracy did not improve from 0.68580
1407/1407 [==============================] - 34s 24ms/step - loss: 0.6022 -
accuracy: 0.7993 - val_loss: 1.0638 - val_accuracy: 0.6622
Epoch 8/100
1407/1407 [==============================] - ETA: 0s - loss: 0.5299 - accuracy:
0.8255
Epoch 00008: val_accuracy improved from 0.68580 to 0.70660, saving model to
results/cifar10/alexnet.weights.best.hdf5
1407/1407 [==============================] - 34s 24ms/step - loss: 0.5299 -
accuracy: 0.8255 - val_loss: 0.9408 - val_accuracy: 0.7066
```

```
Epoch 9/100
1405/1407 [============================>.] - ETA: 0s - loss: 0.4230 - accuracy:
0.8633
Epoch 00009: val_accuracy improved from 0.70660 to 0.73000, saving model to
results/cifar10/alexnet.weights.best.hdf5
1407/1407 [==============================] - 34s 24ms/step - loss: 0.4228 -
accuracy: 0.8634 - val_loss: 0.9134 - val_accuracy: 0.7300
Epoch 10/100
1407/1407 [==============================] - ETA: 0s - loss: 0.3582 - accuracy:
0.8852
Epoch 00010: val_accuracy improved from 0.73000 to 0.73280, saving model to
results/cifar10/alexnet.weights.best.hdf5
1407/1407 [==============================] - 34s 24ms/step - loss: 0.3582 -
accuracy: 0.8852 - val_loss: 0.9120 - val_accuracy: 0.7328
Epoch 11/100
1407/1407 [==============================] - ETA: 0s - loss: 0.3022 - accuracy:
0.9035
Epoch 00011: val_accuracy did not improve from 0.73280
1407/1407 [==============================] - 31s 22ms/step - loss: 0.3022 -
accuracy: 0.9035 - val_loss: 1.0263 - val_accuracy: 0.6922
Epoch 12/100
1407/1407 [==============================] - ETA: 0s - loss: 0.2579 - accuracy:
0.9190
Epoch 00012: val_accuracy did not improve from 0.73280
1407/1407 [==============================] - 30s 21ms/step - loss: 0.2579 -
accuracy: 0.9190 - val_loss: 1.2463 - val_accuracy: 0.6898
Epoch 13/100
1406/1407 [============================>.] - ETA: 0s - loss: 0.3696 - accuracy:
0.8824
Epoch 00013: val_accuracy improved from 0.73280 to 0.74220, saving model to
results/cifar10/alexnet.weights.best.hdf5
1407/1407 [==============================] - 35s 25ms/step - loss: 0.3696 -
accuracy: 0.8824 - val_loss: 0.9610 - val_accuracy: 0.7422
Epoch 14/100
1407/1407 [==============================] - ETA: 0s - loss: 0.1797 - accuracy:
0.9445
Epoch 00014: val_accuracy did not improve from 0.74220
1407/1407 [==============================] - 33s 24ms/step - loss: 0.1797 -
accuracy: 0.9445 - val_loss: 1.2070 - val_accuracy: 0.7098
Epoch 15/100
1405/1407 [============================>.] - ETA: 0s - loss: 0.2776 - accuracy:
0.9139
Epoch 00015: val_accuracy did not improve from 0.74220
1407/1407 [==============================] - 35s 25ms/step - loss: 0.2777 -
accuracy: 0.9139 - val_loss: 1.0512 - val_accuracy: 0.7216
Epoch 16/100
1405/1407 [============================>.] - ETA: 0s - loss: 0.1719 - accuracy:
0.9470
```

```
Epoch 00016: val_accuracy improved from 0.74220 to 0.74620, saving model to
results/cifar10/alexnet.weights.best.hdf5
1407/1407 [==============================] - 36s 25ms/step - loss: 0.1720 -
accuracy: 0.9470 - val_loss: 1.9644 - val_accuracy: 0.7462
Epoch 17/100
1405/1407 [============================>.] - ETA: 0s - loss: 0.1353 - accuracy:
0.9598
Epoch 00017: val_accuracy did not improve from 0.74620
1407/1407 [==============================] - 33s 23ms/step - loss: 0.1352 -
accuracy: 0.9598 - val_loss: 11.1757 - val_accuracy: 0.7320
Epoch 18/100
1407/1407 [==============================] - ETA: 0s - loss: 0.1310 - accuracy:
0.9613
Epoch 00018: val_accuracy did not improve from 0.74620
1407/1407 [==============================] - 33s 24ms/step - loss: 0.1310 -
accuracy: 0.9613 - val_loss: 1.5545 - val_accuracy: 0.6550
Epoch 19/100
1407/1407 [==============================] - ETA: 0s - loss: 0.1294 - accuracy:
0.9624
Epoch 00019: val_accuracy did not improve from 0.74620
1407/1407 [==============================] - 34s 24ms/step - loss: 0.1294 -
accuracy: 0.9624 - val_loss: 1.3648 - val_accuracy: 0.7234
Epoch 20/100
1405/1407 [============================>.] - ETA: 0s - loss: 0.1364 - accuracy:
0.9587
Epoch 00020: val_accuracy did not improve from 0.74620
1407/1407 [==============================] - 34s 24ms/step - loss: 0.1364 -
accuracy: 0.9586 - val_loss: 1.3799 - val_accuracy: 0.7084
Epoch 21/100
1406/1407 [============================>.] - ETA: 0s - loss: 0.1210 - accuracy:
0.9641
Epoch 00021: val_accuracy did not improve from 0.74620
1407/1407 [==============================] - 33s 24ms/step - loss: 0.1211 -
accuracy: 0.9641 - val_loss: 1.3672 - val_accuracy: 0.7190
Epoch 22/100
1405/1407 [============================>.] - ETA: 0s - loss: 0.1133 - accuracy:
0.9662
Epoch 00022: val_accuracy did not improve from 0.74620
1407/1407 [==============================] - 33s 24ms/step - loss: 0.1132 -
accuracy: 0.9663 - val_loss: 1.2889 - val_accuracy: 0.7302
Epoch 23/100
1406/1407 [============================>.] - ETA: 0s - loss: 0.1098 - accuracy:
0.9673
Epoch 00023: val_accuracy did not improve from 0.74620
1407/1407 [==============================] - 35s 25ms/step - loss: 0.1098 -
accuracy: 0.9673 - val_loss: 1.6106 - val_accuracy: 0.7150
Epoch 24/100
1406/1407 [============================>.] - ETA: 0s - loss: 0.1078 - accuracy:
```

```
0.9673
Epoch 00024: val_accuracy did not improve from 0.74620
1407/1407 [==============================] - 35s 25ms/step - loss: 0.1078 -
accuracy: 0.9673 - val_loss: 1.4081 - val_accuracy: 0.7168
Epoch 25/100
1406/1407 [============================>.] - ETA: 0s - loss: 0.0959 - accuracy:
0.9709
Epoch 00025: val_accuracy did not improve from 0.74620
1407/1407 [==============================] - 31s 22ms/step - loss: 0.0959 -
accuracy: 0.9709 - val_loss: 1.7263 - val_accuracy: 0.6644
Epoch 26/100
1407/1407 [==============================] - ETA: 0s - loss: 0.0873 - accuracy:
0.9737
Epoch 00026: val_accuracy did not improve from 0.74620
1407/1407 [==============================] - 34s 24ms/step - loss: 0.0873 -
accuracy: 0.9737 - val_loss: 1.4485 - val_accuracy: 0.7152
```

[65]:
```
pd.DataFrame(alex_history.history)[['accuracy', 'val_accuracy']].
 ↪plot(figsize=(14, 5))
sns.despine();
```



[66]:
```
alexnet.load_weights(alexnet_path)
```

After training for 20 episodes, each of which takes a little under 30 seconds on a single GPU, we obtain 76.84% test accuracy.

[67]:
```
alex_accuracy = alexnet.evaluate(X_test, y_test, verbose=0)[1]
print('Test Accuracy: {:.2%}'.format(alex_accuracy))
```

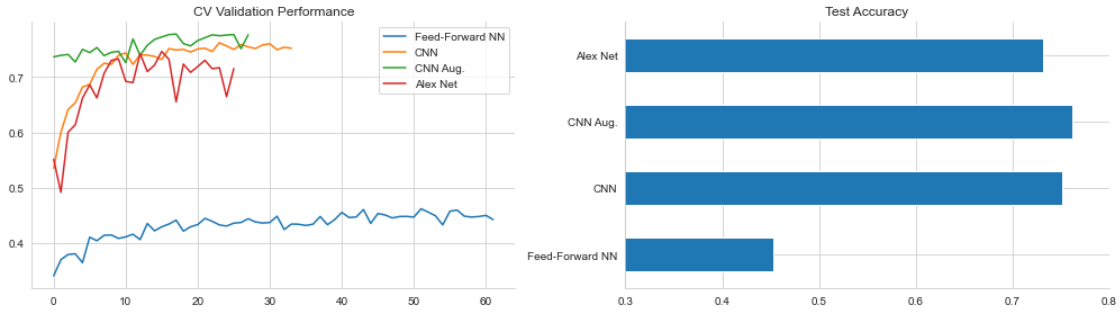
```
Test Accuracy: 73.21%
```

1.7 Compare Results

[68]:
```
cv_results = pd.DataFrame(
    {'Feed-Forward NN': pd.Series(mlp_history.history['val_accuracy']),
        'CNN': pd.Series(cnn_history.history['val_accuracy']),
        'CNN Aug.': pd.Series(cnn_aug_history.history['val_accuracy']),
        'Alex Net': pd.Series(alex_history.history['val_accuracy'])
    })
```

[69]:
```
test_accuracy = pd.Series({
    'Feed-Forward NN': mlp_accuracy,
    'CNN': cnn_accuracy,
    'CNN Aug.': cnn_aug_accuracy,
    'Alex Net': alex_accuracy
})
```

[70]:
```
fig, axes = plt.subplots(ncols=2, figsize=(14, 4))
cv_results.plot(ax=axes[0], title='CV Validation Performance')
test_accuracy.plot.barh(ax=axes[1], xlim=(.3, .8), title='Test Accuracy')
fig.tight_layout()
sns.despine()
fig.savefig(results_path / 'comparison', dpi=300);
```



1.8 TensorBoard visualization

[71]:
```
%load_ext tensorboard
```

[72]:
```
%tensorboard --logdir results/cifar10/logs
```

```
Reusing TensorBoard on port 6009 (pid 11959), started 4:08:40 ago. (Use '!kill␣
↪11959' to kill it.)
```

```
<IPython.core.display.HTML object>
```

[ ]:
```

```