Run VGG16 on 30 classes without tuning

In this part, we loads all qualified data (30 classes) in VGG16 and only add an output layer.

However, based on accuracy and loss curves, we can clearly see that there exists severe overfitting. To address overfiting, adding noises, regularization may help.

Obviously, we need to find out how we add layers can reduce overfitting as much as possible and maintain the accuracy at the same time.

Therefore, we decide to use 3 classes which contains most paintings and 20 epochs for figuring out how we can train a better model. And then, we go back to 30 classes and compare the results with this untuning version.

In [1]:
```
import numpy as np
import os
from tensorflow.keras import applications
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import optimizers
from tensorflow.keras.models import Sequential
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dropout, Flatten, Dense
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
```

Validation of using GPU

In [2]:
```
from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())
```

```
[name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality {
}
incarnation: 16556785846815173339
, name: "/device:GPU:0"
device_type: "GPU"
memory_limit: 2264907776
locality {
  bus_id: 1
  links {
  }
}
incarnation: 15477623663351877341
physical_device_desc: "device: 0, name: GeForce GTX 970M, pci bus id: 000
0:01:00.0, compute capability: 5.2"
]
```

Loading the pre-trained VGG16

In [3]:
```
import tensorflow.keras.backend as K
K.clear_session()
```

In [4]:
```
nrow = 200
ncol = 200
base_model = applications.VGG16(weights='imagenet', input_shape=(nrow,ncol,3
model = Sequential()

for layer in base_model.layers:
    model.add(layer)
for layer in model.layers:
    layer.trainable = False
```

```
WARNING:tensorflow:From D:\Anaconda\envs\tensorflow\lib\site-packages\ten
sorflow\python\ops\resource_variable_ops.py:435: colocate_with (from tens
orflow.python.framework.ops) is deprecated and will be removed in a futur
e version.
Instructions for updating:
Colocations handled automatically by placer.
```

Now, we only add a final fully-connected layer. Since this is a multiple classification, there should be 30 output and softmax activation.

In [5]:
```
model.add(Flatten())
model.add(Dense(30, activation = 'softmax'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
block1_conv1 (Conv2D)        (None, 200, 200, 64)      1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 200, 200, 64)      36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 100, 100, 64)      0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 100, 100, 128)     73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 100, 100, 128)     147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 50, 50, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 50, 50, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 50, 50, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 50, 50, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 25, 25, 256)       0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 25, 25, 512)       1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 25, 25, 512)       2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 25, 25, 512)       2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 12, 12, 512)       0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 12, 12, 512)       2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 12, 12, 512)       2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 12, 12, 512)       2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 6, 6, 512)         0         
_________________________________________________________________
flatten (Flatten)            (None, 18432)             0         
_________________________________________________________________
dense (Dense)                (None, 30)                552990    
=================================================================
Total params: 15,267,678
Trainable params: 552,990
Non-trainable params: 14,714,688
_________________________________________________________________
```

Using Generators to Load Data

In [6]:
```
train_data_dir = './images_train'
batch_size = 32
train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)
train_generator = train_datagen.flow_from_directory(
                        train_data_dir,
                        target_size=(nrow,ncol),
                        batch_size=batch_size,
                        class_mode='categorical')
```

```
Found 5687 images belonging to 30 classes.
```

In [7]:
```
test_data_dir = './images_test'
batch_size = 32
test_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)
test_generator = train_datagen.flow_from_directory(
                        test_data_dir,
                        target_size=(nrow,ncol),
                        batch_size=batch_size,
                        class_mode='categorical')
```

```
Found 1404 images belonging to 30 classes.
```

Train the model

Compile the model. we are performing multiple classification, so we use 'categorical_crossentropy' loss function.

In [8]:
```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['a
steps_per_epoch = train_generator.n // batch_size
validation_steps = test_generator.n // batch_size
```

Now, we run the fit. Since we run 120 epochs, even with GPU, it will take hours (about 4 hours in our case).

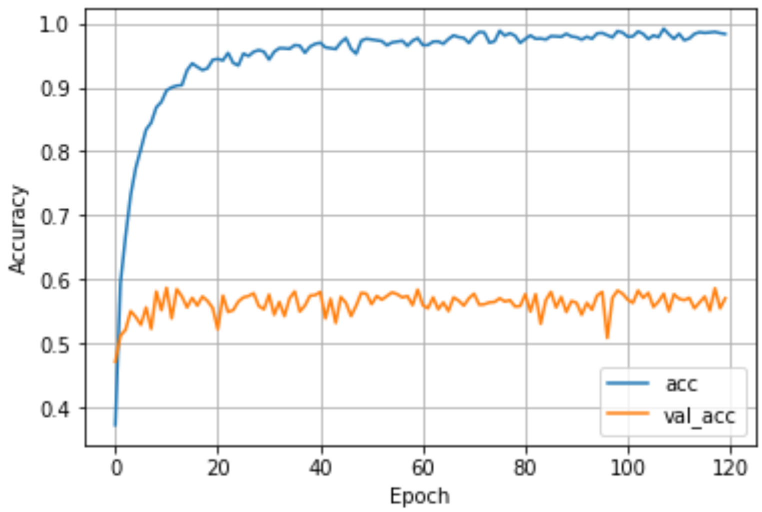
In [9]:
```
nepochs = 120  # Number of epochs

# Call the fit_generator function
hist = model.fit_generator(
    train_generator,
    epochs=nepochs,
    steps_per_epoch=steps_per_epoch,
    validation_data=test_generator,
    validation_steps=validation_steps)
```

```
WARNING:tensorflow:From D:\Anaconda\envs\tensorflow\lib\site-packages\ten
sorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.python.op
s.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/120
44/44 [==============================] - 36s 822ms/step - loss: 1.8798 -
acc: 0.4715
178/178 [==============================] - 183s 1s/step - loss: 2.3449 -
acc: 0.3726 - val_loss: 1.8798 - val_acc: 0.4715
Epoch 2/120
44/44 [==============================] - 29s 664ms/step - loss: 1.8880 -
acc: 0.5121
178/178 [==============================] - 133s 748ms/step - loss: 1.4229
- acc: 0.5936 - val_loss: 1.8880 - val_acc: 0.5121
Epoch 3/120
44/44 [==============================] - 30s 677ms/step - loss: 1.8866 -
acc: 0.5214
178/178 [==============================] - 143s 805ms/step - loss: 1.1424
```

Plot the accuracy curve

In [10]:
```
hist_his = hist.history
acc = hist_his['acc']
val_acc = hist_his['val_acc']
plt.plot(acc)
plt.plot(val_acc)
plt.grid()
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['acc','val_acc'], loc = 4)
```

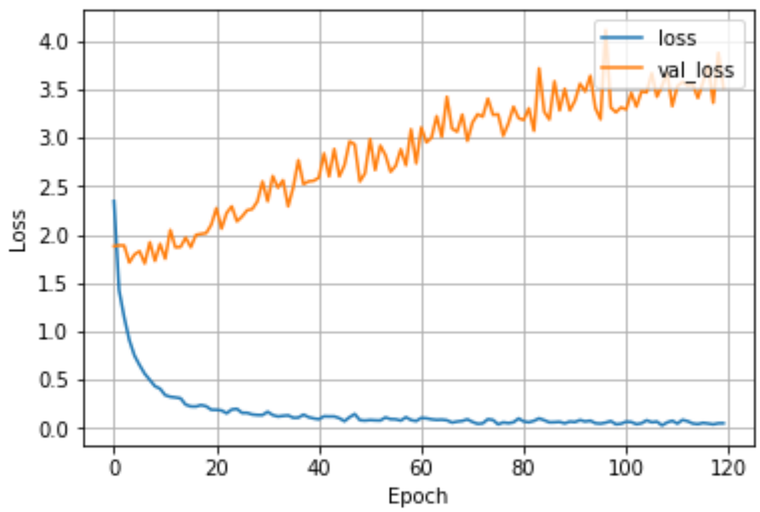
Out[10]: <matplotlib.legend.Legend at 0x2d9317ccb00>



Plot the loss curve

In [11]:
```
loss = hist_his['loss']
val_loss = hist_his['val_loss']
plt.plot(loss)
plt.plot(val_loss)
plt.grid()
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['loss','val_loss'], loc = 1)
```

Out[11]: <matplotlib.legend.Legend at 0x2d916831358>



In [12]:
```
print(loss)
print(val_loss)
print(acc)
print(val_acc)
```

```
[2.3461001035050724, 1.4235979016874845, 1.1430863561457525, 0.9086284632
525168, 0.7497734822894895, 0.6507400545521362, 0.5618185450956527, 0.497
515937167688, 0.4333582236034413, 0.4072008218772805, 0.340391851695660 9,
0.32332254938371197, 0.3169648504349434, 0.30803932771492826, 0.242359506
10026422, 0.2240992051311 6352, 0.2229919945112 0705, 0.238132092828582 5,
0.22565900431023933, 0.1895868973198938, 0.18879720360218838, 0.182640925
18734307, 0.1530688407074143 3, 0.1924067095972771 8, 0.1987025859746548 2,
0.15718773798697733, 0.15746098371291067, 0.14063597524793126, 0.13533142
471598555, 0.13465000178611863, 0.16852534865377153, 0.13479207398729207,
0.11998056702503153, 0.12652170437751514, 0.1314194848541847, 0.106885104
4963235, 0.10811103968444641, 0.13935735310389033, 0.11262571445435511,
0.09994827654374512, 0.09156880270822702, 0.12170841077027221, 0.11986629
047051284, 0.12042203829077608, 0.10257396470124522, 0.07141154148305237,
0.11286345072888482, 0.14545853183142474, 0.08261463096709563, 0.07816871
711160547, 0.0834815292300649, 0.08061120286518395, 0.07870132581194748,
0.1100104708078458, 0.09237889285474958, 0.09098280840773605, 0.077585979
0803553, 0.11242035616805647, 0.08350875193663551, 0.07306732858002385,
0.10495222422125806, 0.10121314234917625, 0.09041668099852507, 0.08502138
9281652, 0.08709571150535855, 0.08119573050985286, 0.05720834045503724,
0.06752291287102116, 0.07401366609014232, 0.09078182027614128, 0.06360876
416197629, 0.04408281449970701, 0.04912982641352288, 0.09239701398996975,
0.08397759706717353, 0.03846376826959649, 0.05900241308116014, 0.05116836
12821664, 0.06045130762034617, 0.09908614304464973, 0.06853826708419579,
0.06010749900452524, 0.07500018046391 63, 0.10037270990330205, 0.081171884
01919835, 0.05964600242207134, 0.06097916190592559, 0.06448622915978296,
0.049291680894329615, 0.06723989536866208, 0.06261745889440076, 0.0828949
253097588, 0.06829598370735772, 0.07648963783003805, 0.05037049173159812
6, 0.04628097680067349, 0.05635536908192869, 0.07244983570611244, 0.03961
103102834645, 0.04535775429545275, 0.06595275171501659, 0.062991343934500
97, 0.0404003969792762, 0.0501003786742999, 0.08006950906923796, 0.060235
21859767723, 0.06697451766250852, 0.028666988472729863, 0.058501245508568
85, 0.07587163815681296, 0.04663811410420758, 0.08466476121967642, 0.0710
8294285135548, 0.04754935567612289, 0.04140392171914331, 0.05381673431490
354, 0.04729886747521981, 0.03891590693035378, 0.049779767958332374, 0.05
005639105934027]
[1.879849160259420 3, 1.8879662372849204, 1.8866435939615422, 1.7108222733
84441, 1.794157469814474, 1.8281896331093528, 1.701570169492201 4, 1.92040
62115062366, 1.728152844038876 7, 1.9042668640613556, 1.75035000389272 5,
2.046810150146484 4, 1.8679933900182897, 1.871904801238667, 1.969529035416
5165, 1.8692320016297428, 1.9964542416009037, 2.0064142779870466, 2.01484
8356897181, 2.101751360026273, 2.273027856241573, 2.0610892826860603, 2.2
23997294902801 5, 2.291876866058869 8, 2.136060793291439, 2.18583698435263
2, 2.2514450224963 1, 2.263073284517635, 2.3427226055752146, 2.54858815399
0832, 2.344073311849074, 2.6048515899614855, 2.4827307760715485, 2.561436
5718581458, 2.291786703196439, 2.488292637196454, 2.7692495503208856, 2.5
21276568824595, 2.5499190959063442, 2.5560632808641954, 2.58809532902457
5, 2.838707988912409, 2.600795794617046, 2.884335707534443, 2.60051744363
5247, 2.7178437899459493, 2.962112814188003 5, 2.93234098228541 3, 2.548693
830316717, 2.631456044587222, 2.986494557424025, 2.6665902733802795, 2.92
0082558285106 4, 2.821283047849482, 2.6478641114451666, 2.71098004687916 1
6, 2.8821339241482993, 2.7162570005113427, 3.0881751017137007, 2.73587841
3373774, 3.1072994470596313, 2.9527191438458185, 3.0017079412937164, 3.22
```

```
37656604159963, 3.0099908736619083, 3.4242454875599253, 3.09478339011018
9, 3.062528184869073, 3.2398743548176507, 2.968336186625741, 3.1587945737
622003, 3.2433025484735314, 3.218614635142413, 3.4046757139942865, 3.2372
12760881944, 3.2428165620023552, 3.021511275659908, 3.1484800306233494,
3.3216919844800774, 3.2022319463166324, 3.184491596438668, 3.305086414922
3673, 3.0715756795623084, 3.7181684239344164, 3.2697226892818105, 3.18903
4489068118, 3.5838129466230217, 3.282617674632506, 3.5069586146961558, 3.
2830505316907708, 3.3833330761302602, 3.559149671684612, 3.47740050879391
75, 3.6409821293570777, 3.3053645572879096, 3.1910788254304365, 4.1142676
31010576, 3.3127417726950212, 3.2631784812970595, 3.313422804529017, 3.29
3671580878171, 3.4643251408230173, 3.3222834400155326, 3.479178431359204
5, 3.4695660200985996, 3.667523671280254, 3.4244197715412485, 3.535086276
856336, 3.6762977242469788, 3.326822979883714, 3.5333000042221765, 3.5777
104551141914, 3.538788849657232, 3.5542741905559194, 3.4110266024416145,
3.5744017741896887, 3.7527989582581953, 3.361420116641305, 3.878032784570
347, 3.514143545519222]
[0.3726042, 0.5936346, 0.66731143, 0.73184454, 0.77474946, 0.803763, 0.83
40074, 0.8447336, 0.86882365, 0.87726396, 0.8951996, 0.8999472, 0.9025848
5, 0.903464, 0.92667484, 0.9374011, 0.93195003, 0.92685074, 0.92983997,
0.94285214, 0.9444347, 0.9419729, 0.9537542, 0.93810445, 0.93406016, 0.95
26991, 0.94883066, 0.9555126, 0.95815015, 0.9556884, 0.9432038, 0.955512
6, 0.96113944, 0.9609636, 0.9600844, 0.9658871, 0.9650079, 0.95393, 0.963
6012, 0.96782136, 0.96957976, 0.96237034, 0.96113944, 0.9597327, 0.970107
26, 0.976965, 0.9600844, 0.95234746, 0.9729207, 0.97590995, 0.9746791, 0.
9734482, 0.9722173, 0.9655354, 0.97010726, 0.97133815, 0.9727449, 0.96465
623, 0.9727449, 0.9767892, 0.9658871, 0.96606296, 0.97133815, 0.9716898,
0.9679972, 0.97538245, 0.98118514, 0.97837174, 0.97714084, 0.9692281, 0.9
796026, 0.9864603, 0.985757, 0.96957976, 0.97186565, 0.98839456, 0.980657
64, 0.9845261, 0.98065764, 0.9694039, 0.9757341, 0.9810093, 0.97590995,
0.9762617, 0.9746791, 0.9799543, 0.9796026, 0.97889924, 0.9836469, 0.9794
2674, 0.9781959, 0.9745033, 0.9790751, 0.9757341, 0.98417443, 0.98487777,
0.981361, 0.9780201, 0.98786706, 0.9854053, 0.97889924, 0.9796026, 0.9873
3956, 0.9834711, 0.97538245, 0.98118514, 0.9781959, 0.9917355, 0.9829435
3, 0.9757341, 0.9836469, 0.97362405, 0.9762617, 0.9834711, 0.98610866, 0.
98505366, 0.985757, 0.98663616, 0.98470193, 0.98329526]
[0.47150996, 0.51210827, 0.52136755, 0.5505698, 0.54131055, 0.5292023, 0.
5562678, 0.52279204, 0.5811966, 0.5519943, 0.5868946, 0.5391738, 0.584757
86, 0.5733618, 0.5562678, 0.57122505, 0.5591168, 0.5733618, 0.5662393, 0.
5555556, 0.52207977, 0.5747863, 0.5491453, 0.5519943, 0.5655271, 0.571937
3, 0.5740741, 0.57834756, 0.55840456, 0.5534188, 0.5762108, 0.5448718, 0.
5648148, 0.54273504, 0.57051283, 0.5811966, 0.54985756, 0.5591168, 0.5747
863, 0.5754986, 0.58048433, 0.5391738, 0.56980056, 0.53205127, 0.5726496,
0.5633903, 0.54273504, 0.5591168, 0.57905984, 0.57763535, 0.56125355, 0.5
740741, 0.56837606, 0.5740741, 0.57977206, 0.5769231, 0.5719373, 0.574074
1, 0.55982906, 0.5840456, 0.55982906, 0.5548433, 0.5719373, 0.5534188, 0.
5641026, 0.5505698, 0.5719373, 0.5669516, 0.5591168, 0.57051283, 0.577635
35, 0.56054133, 0.56125355, 0.5641026, 0.5648148, 0.57051283, 0.5655271,
0.5676638, 0.5576923, 0.55840456, 0.5769231, 0.54985756, 0.5769231, 0.530
6268, 0.5676638, 0.58048433, 0.5562678, 0.5726496, 0.5491453, 0.5662393,
0.5633903, 0.5448718, 0.5633903, 0.55270654, 0.5747863, 0.58048433, 0.508
547, 0.57122505, 0.5826211, 0.57763535, 0.56837606, 0.5633903, 0.5826211,
0.57122505, 0.57905984, 0.5569801, 0.5655271, 0.57763535, 0.5505698, 0.57
69231, 0.56980056, 0.5676638, 0.57122505, 0.5548433, 0.5641026, 0.573361
8, 0.55128205, 0.58618236, 0.5548433, 0.57051283]
```

Summary

Based on above accuracy and loss curves, we can clearly see that there exists severe overfitting. Obviously, we need to find out how we add layers can reduce overfitting as much as possible and maintain the accuracy at the same time.

So we decide to use 3 classes which contains most paintings and 20 epochs as the starting point. To address overfiting, adding noises, regularization may help.

In [ ]:

```

```

Run VGG16 on selected 3 classes without tuning

In this part, we only loads 3 classes with most paintings in VGG16 and still without tuning, because we need to compare it with the other tuned versions.

Though this time we only run 20 epochs, we can still tell there exists severe overfitting.

In [1]:
```
import numpy as np
import os
from tensorflow.keras import applications
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import optimizers
from tensorflow.keras.models import Sequential
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dropout, Flatten, Dense
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())
```

```
[name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality {
}
incarnation: 8866674015826737595
, name: "/device:GPU:0"
device_type: "GPU"
memory_limit: 2264907776
locality {
  bus_id: 1
  links {
  }
}
incarnation: 10400245706179158261
physical_device_desc: "device: 0, name: GeForce GTX 970M, pci bus id: 000
0:01:00.0, compute capability: 5.2"
]
```

In [3]:
```
import tensorflow.keras.backend as K
K.clear_session()
```

In [4]:
```
nrow = 200
ncol = 200
base_model = applications.VGG16(weights='imagenet', input_shape=(nrow,ncol,3
model = Sequential()

for layer in base_model.layers:
    model.add(layer)
for layer in model.layers:
    layer.trainable = False
```

```
WARNING:tensorflow:From D:\Anaconda\envs\tensorflow\lib\site-packages\ten
sorflow\python\ops\resource_variable_ops.py:435: colocate_with (from tens
orflow.python.framework.ops) is deprecated and will be removed in a futur
e version.
Instructions for updating:
Colocations handled automatically by placer.
```

In [5]:
```
model.add(Flatten())
model.add(Dense(3, activation = 'softmax'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
block1_conv1 (Conv2D)        (None, 200, 200, 64)      1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 200, 200, 64)      36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 100, 100, 64)      0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 100, 100, 128)     73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 100, 100, 128)     147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 50, 50, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 50, 50, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 50, 50, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 50, 50, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 25, 25, 256)       0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 25, 25, 512)       1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 25, 25, 512)       2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 25, 25, 512)       2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 12, 12, 512)       0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 12, 12, 512)       2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 12, 12, 512)       2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 12, 12, 512)       2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 6, 6, 512)         0         
_________________________________________________________________
flatten (Flatten)            (None, 18432)             0         
_________________________________________________________________
dense (Dense)                (None, 3)                 55299     
=================================================================
Total params: 14,769,987
Trainable params: 55,299
Non-trainable params: 14,714,688
_________________________________________________________________
```

In [6]:
```
train_data_dir = './images_train'
batch_size = 32
train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)
train_generator = train_datagen.flow_from_directory(
                        train_data_dir,
                        target_size=(nrow,ncol),
                        batch_size=batch_size,
                        class_mode='categorical')
```

```
Found 1616 images belonging to 3 classes.
```

In [7]:
```
test_data_dir = './images_test'
batch_size = 32
test_datagen = ImageDataGenerator(rescale=1./255,
                                  shear_range=0.2,
                                  zoom_range=0.2,
                                  horizontal_flip=True)
test_generator = train_datagen.flow_from_directory(
                        test_data_dir,
                        target_size=(nrow,ncol),
                        batch_size=batch_size,
                        class_mode='categorical')
```

```
Found 402 images belonging to 3 classes.
```

In [8]:
```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['a
steps_per_epoch = train_generator.n // batch_size
validation_steps = test_generator.n // batch_size
```

In [9]:
```
nepochs = 20  # Number of epochs

# Call the fit_generator function
hist = model.fit_generator(
    train_generator,
    epochs=nepochs,
    steps_per_epoch=steps_per_epoch,
    validation_data=test_generator,
    validation_steps=validation_steps)
```

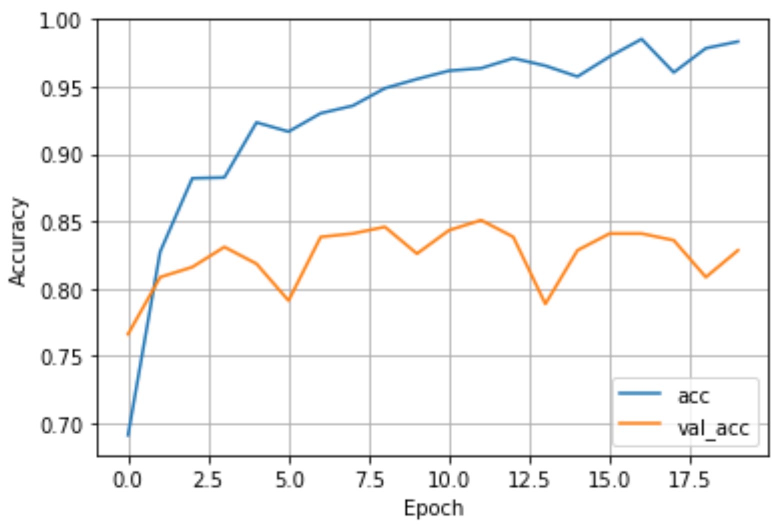
```
WARNING:tensorflow:From D:\Anaconda\envs\tensorflow\lib\site-packages\ten
sorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.python.op
s.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/20
13/13 [==============================] - 13s 975ms/step - loss: 0.5347 -
acc: 0.7662
51/51 [==============================] - 48s 951ms/step - loss: 0.7488 -
acc: 0.6912 - val_loss: 0.5347 - val_acc: 0.7662
Epoch 2/20
13/13 [==============================] - 11s 884ms/step - loss: 0.4963 -
acc: 0.8085
51/51 [==============================] - 41s 803ms/step - loss: 0.4483 -
acc: 0.8274 - val_loss: 0.4963 - val_acc: 0.8085
Epoch 3/20
13/13 [==============================] - 11s 879ms/step - loss: 0.5242 -
acc: 0.8159
51/51 [==============================] - 41s 801ms/step - loss: 0.3342 -
acc: 0.8818 - val_loss: 0.5242 - val_acc: 0.8159
Epoch 4/20
13/13 [==============================] - 13s 1s/step - loss: 0.4377 - ac
c: 0.8308
51/51 [==============================] - 43s 848ms/step - loss: 0.3334 -
acc: 0.8824 - val_loss: 0.4377 - val_acc: 0.8308
Epoch 5/20
13/13 [==============================] - 13s 1s/step - loss: 0.4771 - ac
c: 0.8184
51/51 [==============================] - 42s 832ms/step - loss: 0.2429 -
acc: 0.9233 - val_loss: 0.4771 - val_acc: 0.8184
Epoch 6/20
13/13 [==============================] - 11s 851ms/step - loss: 0.5297 -
acc: 0.7910
51/51 [==============================] - 44s 864ms/step - loss: 0.2409 -
acc: 0.9165 - val_loss: 0.5297 - val_acc: 0.7910
Epoch 7/20
13/13 [==============================] - 10s 773ms/step - loss: 0.4441 -
acc: 0.8383
51/51 [==============================] - 38s 750ms/step - loss: 0.2070 -
acc: 0.9301 - val_loss: 0.4441 - val_acc: 0.8383
Epoch 8/20
13/13 [==============================] - 10s 781ms/step - loss: 0.4205 -
acc: 0.8408
51/51 [==============================] - 38s 746ms/step - loss: 0.1901 -
acc: 0.9356 - val_loss: 0.4205 - val_acc: 0.8408
Epoch 9/20
13/13 [==============================] - 11s 829ms/step - loss: 0.4478 -
```

```
acc: 0.8458
51/51 [==============================] - 38s 745ms/step - loss: 0.1703 -
acc: 0.9486 - val_loss: 0.4478 - val_acc: 0.8458
Epoch 10/20
13/13 [==============================] - 12s 953ms/step - loss: 0.4054 -
acc: 0.8259
51/51 [==============================] - 41s 803ms/step - loss: 0.1497 -
acc: 0.9554 - val_loss: 0.4054 - val_acc: 0.8259
Epoch 11/20
13/13 [==============================] - 11s 827ms/step - loss: 0.4796 -
acc: 0.8433
51/51 [==============================] - 38s 739ms/step - loss: 0.1369 -
acc: 0.9616 - val_loss: 0.4796 - val_acc: 0.8433
Epoch 12/20
13/13 [==============================] - 11s 882ms/step - loss: 0.3918 -
acc: 0.8507
51/51 [==============================] - 40s 779ms/step - loss: 0.1280 -
acc: 0.9635 - val_loss: 0.3918 - val_acc: 0.8507
Epoch 13/20
13/13 [==============================] - 11s 869ms/step - loss: 0.4435 -
acc: 0.8383
51/51 [==============================] - 38s 737ms/step - loss: 0.1169 -
acc: 0.9709 - val_loss: 0.4435 - val_acc: 0.8383
Epoch 14/20
13/13 [==============================] - 12s 923ms/step - loss: 0.5926 -
acc: 0.7886
51/51 [==============================] - 40s 778ms/step - loss: 0.1165 -
acc: 0.9653 - val_loss: 0.5926 - val_acc: 0.7886
Epoch 15/20
13/13 [==============================] - 11s 884ms/step - loss: 0.4586 -
acc: 0.8284
51/51 [==============================] - 39s 774ms/step - loss: 0.1200 -
acc: 0.9573 - val_loss: 0.4586 - val_acc: 0.8284
Epoch 16/20
13/13 [==============================] - 13s 1s/step - loss: 0.4632 - ac
c: 0.8408
51/51 [==============================] - 44s 866ms/step - loss: 0.1036 -
acc: 0.9722 - val_loss: 0.4632 - val_acc: 0.8408
Epoch 17/20
13/13 [==============================] - 12s 939ms/step - loss: 0.4997 -
acc: 0.8408
51/51 [==============================] - 44s 855ms/step - loss: 0.0895 -
acc: 0.9851 - val_loss: 0.4997 - val_acc: 0.8408
Epoch 18/20
13/13 [==============================] - 10s 780ms/step - loss: 0.4993 -
acc: 0.8358
51/51 [==============================] - 37s 726ms/step - loss: 0.1238 -
acc: 0.9604 - val_loss: 0.4993 - val_acc: 0.8358
Epoch 19/20
13/13 [==============================] - 10s 803ms/step - loss: 0.5603 -
acc: 0.8085
51/51 [==============================] - 42s 823ms/step - loss: 0.0796 -
acc: 0.9783 - val_loss: 0.5603 - val_acc: 0.8085
Epoch 20/20
13/13 [==============================] - 10s 781ms/step - loss: 0.4830 -
acc: 0.8284
```

```
51/51 [==============================] - 38s 750ms/step - loss: 0.0806 -
acc: 0.9833 - val_loss: 0.4830 - val_acc: 0.8284
```

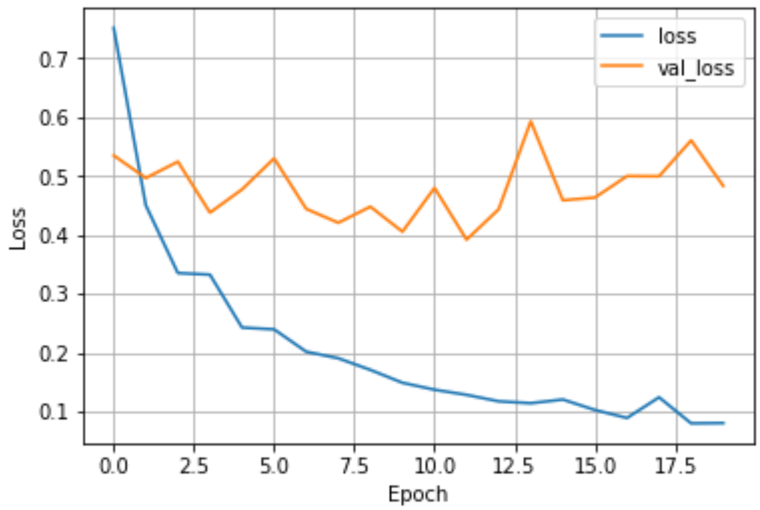
In [10]:
```
hist_his = hist.history
acc = hist_his['acc']
val_acc = hist_his['val_acc']
plt.plot(acc)
plt.plot(val_acc)
plt.grid()
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['acc','val_acc'], loc = 4)
```

Out[10]:
```
<matplotlib.legend.Legend at 0x1445d9b44e0>
```



In [11]:
```
loss = hist_his['loss']
val_loss = hist_his['val_loss']
plt.plot(loss)
plt.plot(val_loss)
plt.grid()
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['loss','val_loss'], loc = 1)
```

Out[11]: <matplotlib.legend.Legend at 0x14476bee470>



In [12]:
```
print(loss)
print(val_loss)
print(acc)
print(val_acc)
```

```
[0.7509350723559314, 0.4501491076875441, 0.33513500932419654, 0.332461600
8465833, 0.24263272545125225, 0.23967051196216357, 0.20171235774708265,
0.19057435534968234, 0.1708688954315563, 0.14899897678653792, 0.137203000
22770862, 0.12850487748585124, 0.11753369001026201, 0.11433112636060998,
0.12057178120801945, 0.10282738853504161, 0.08929582699985787, 0.12433314
500468792, 0.08017670243314587, 0.08062718273832066]
[0.5346578520077926, 0.49634360120846677, 0.5241671640139359, 0.437715965
96791196, 0.47711492157899416, 0.529675291134761, 0.44406301700151884, 0.
4204594859710106, 0.4477960444413699, 0.4053718986419531, 0.4796006221037
645, 0.3917713027734023, 0.44345738566838777, 0.5926056137451758, 0.45863
019273831296, 0.46321375782673174, 0.4996836426166388, 0.499333129479334
9, 0.5602593078063085, 0.4830422126329862]
[0.6912129, 0.8273515, 0.8818069, 0.8824257, 0.9232673, 0.9164604, 0.9300
743, 0.93564355, 0.9486386, 0.9554455, 0.9616337, 0.9634901, 0.97091585,
0.9653465, 0.957302, 0.9721535, 0.9851485, 0.96039605, 0.9783416, 0.98329
21]
[0.76616913, 0.80845773, 0.8159204, 0.8308458, 0.81840795, 0.7910448, 0.8
3830845, 0.840796, 0.84577113, 0.82587063, 0.8432836, 0.8507463, 0.838308
45, 0.78855723, 0.82835823, 0.840796, 0.840796, 0.8358209, 0.80845773, 0.
82835823]
```

In [ ]:

Run VGG16 on 3 classes test1

For this part, we loads 3 classes with most paintings in VGG16 and adds several layers to test the performance. Still, run 20 epochs.

Before adding layers, we let the `base_model.output` load into variable `x`. Then, we just operate on the `x`. The operations are as following:

- A `Flatten()(x)` layer which reshapes the outputs to a single channel.
- A fully-connected layer with 2304 output units and `relu` activation.
- A `GaussianNoise(0.1)(x)` layer.
- A `Dropout(0.5)(x)` layer.
- A fully-connected layer with 288 output units and `relu` activation.
- A `BatchNormalization()(x)` layer.
- A `Dropout(0.5)(x)` layer.
- A final fully-connected layer. Since this is a multiple classification, there should be three output and `softmax` activation. To mitigate overfitting, we add several arguments: `kernel_initializer='random_uniform'`, `bias_initializer='random_uniform'`, and `bias_regularizer=regularizers.l2(0.01)`.

However, at the end of this test1, we can still clearly see overfitting.

In [1]:
```
import numpy as np
import os
from tensorflow.keras import applications
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import optimizers, regularizers
from tensorflow.keras.models import Sequential
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dropout, Flatten, Dense, GaussianNoise,
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())
```

```
[name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality {
}
incarnation: 13973592982641359775
, name: "/device:XLA_GPU:0"
device_type: "XLA_GPU"
memory_limit: 17179869184
locality {
}
incarnation: 15866717560866743607
physical_device_desc: "device: XLA_GPU device"
, name: "/device:XLA_CPU:0"
device_type: "XLA_CPU"
memory_limit: 17179869184
locality {
}
incarnation: 17454529404844390179
physical_device_desc: "device: XLA_CPU device"
, name: "/device:GPU:0"
device_type: "GPU"
memory_limit: 15856546612
locality {
  bus_id: 1
  links {
  }
}
incarnation: 11389401770462418927
physical_device_desc: "device: 0, name: Tesla P100-PCIE-16GB, pci bus id:
0000:00:04.0, compute capability: 6.0"
]
```

In [3]:
```
import tensorflow.keras.backend as K
K.clear_session()
```

Before adding layers, we let the `base_model.output` load into variable `x`. Then, we just operate on the `x`. The operations are as following:

- A `Flatten()(x)` layer which reshapes the outputs to a single channel.
- A fully-connected layer with 2304 output units and `relu` activation.
- A `GaussianNoise(0.1)(x)` layer.
- A `Dropout(0.5)(x)` layer.
- A fully-connected layer with 288 output units and `relu` activation.
- A `BatchNormalization()(x)` layer.
- A `Dropout(0.5)(x)` layer.
- A final fully-connected layer. Since this is a multiple classification, there should be three output and `softmax` activation. To mitigate overfitting, we add several arguments: `kernel_initializer='random_uniform'`, `bias_initializer='random_uniform'`, and `bias_regularizer=regularizers.l2(0.01)`.

In [1]:
```
nrow = 200
ncol = 200
nclass = 3
base_model = applications.VGG16(weights='imagenet', input_shape=(nrow,ncol,3
for layer in base_model.layers:
    layer.trainable = False

x = base_model.output
x = Flatten()(x)
x = Dense(2304, activation = 'relu')(x) # 18432/4
x = GaussianNoise(0.1)(x) # add noise to mitigate overfitting (regularizatio
x = Dropout(0.5)(x)
x = Dense(288, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)
pred = Dense(nclass, activation='softmax',
             kernel_initializer='random_uniform',
             bias_initializer='random_uniform',
             bias_regularizer=regularizers.l2(0.01),
             name='predictions')(x)
model = Model(inputs=base_model.input, outputs=pred)
```

```
---------------------------------------------------------------------------
--
NameError                                 Traceback (most recent call las
t)
<ipython-input-1-ff2faead8ef7> in <module>()
      2 ncol = 200
      3 nclass = 3
----> 4 base_model = applications.VGG16(weights='imagenet', input_shape=(
nrow,ncol,3), include_top=False)
      5 for layer in base_model.layers:
      6     layer.trainable = False

NameError: name 'applications' is not defined
```

In [5]:
```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         (None, 200, 200, 3)       0         
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 200, 200, 64)      1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 200, 200, 64)      36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 100, 100, 64)      0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 100, 100, 128)     73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 100, 100, 128)     147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 50, 50, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 50, 50, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 50, 50, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 50, 50, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 25, 25, 256)       0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 25, 25, 512)       1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 25, 25, 512)       2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 25, 25, 512)       2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 12, 12, 512)       0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 12, 12, 512)       2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 12, 12, 512)       2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 12, 12, 512)       2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 6, 6, 512)         0         
_________________________________________________________________
flatten (Flatten)            (None, 18432)             0         
_________________________________________________________________
dense (Dense)                (None, 2304)              42469632  
_________________________________________________________________
gaussian_noise (GaussianNois (None, 2304)              0         
_________________________________________________________________
dropout (Dropout)            (None, 2304)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 288)               663840    
_________________________________________________________________
batch_normalization_v1 (Batc (None, 288)               1152      
_________________________________________________________________
dropout_1 (Dropout)          (None, 288)               0         
_________________________________________________________________
```

```
predictions (Dense)          (None, 3)                 867
=================================================================
Total params: 57,850,179
Trainable params: 43,134,915
Non-trainable params: 14,715,264
_________________________________________________________________
```

In [6]:
```
train_data_dir = './images_train'
batch_size = 32
train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)
train_generator = train_datagen.flow_from_directory(
                        train_data_dir,
                        target_size=(nrow,ncol),
                        batch_size=batch_size,
                        class_mode='categorical')
```

```
Found 1616 images belonging to 3 classes.
```

In [7]:
```
test_data_dir = './images_test'
batch_size = 32
test_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)
test_generator = train_datagen.flow_from_directory(
                        test_data_dir,
                        target_size=(nrow,ncol),
                        batch_size=batch_size,
                        class_mode='categorical')
```

```
Found 402 images belonging to 3 classes.
```

In [8]:
```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['a
steps_per_epoch = train_generator.n // batch_size
validation_steps = test_generator.n // batch_size
```

In [9]:
```
nepochs = 20  # Number of epochs

# Call the fit_generator function
hist = model.fit_generator(
    train_generator,
    epochs=nepochs,
    steps_per_epoch=steps_per_epoch,
    validation_data=test_generator,
    validation_steps=validation_steps)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.5/dist-packages/tensorflo
w/python/ops/math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_
ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/20
13/13 [==============================] - 29s 2s/step - loss: 1.2578 - ac
c: 0.5821
51/51 [==============================] - 103s 2s/step - loss: 0.9238 - ac
c: 0.5811 - val_loss: 1.2578 - val_acc: 0.5821
Epoch 2/20
13/13 [==============================] - 29s 2s/step - loss: 0.6700 - ac
c: 0.7413
51/51 [==============================] - 88s 2s/step - loss: 0.6401 - ac
c: 0.7401 - val_loss: 0.6700 - val_acc: 0.7413
Epoch 3/20
13/13 [==============================] - 28s 2s/step - loss: 0.5407 - ac
c: 0.7836
51/51 [==============================] - 86s 2s/step - loss: 0.5565 - ac
c: 0.7642 - val_loss: 0.5407 - val_acc: 0.7836
Epoch 4/20
13/13 [==============================] - 28s 2s/step - loss: 0.5485 - ac
c: 0.8010
51/51 [==============================] - 87s 2s/step - loss: 0.4907 - ac
c: 0.7995 - val_loss: 0.5485 - val_acc: 0.8010
Epoch 5/20
13/13 [==============================] - 28s 2s/step - loss: 0.5116 - ac
c: 0.7910
51/51 [==============================] - 87s 2s/step - loss: 0.4413 - ac
c: 0.8280 - val_loss: 0.5116 - val_acc: 0.7910
Epoch 6/20
13/13 [==============================] - 29s 2s/step - loss: 0.4602 - ac
c: 0.8234
51/51 [==============================] - 88s 2s/step - loss: 0.4384 - ac
c: 0.8348 - val_loss: 0.4602 - val_acc: 0.8234
Epoch 7/20
13/13 [==============================] - 29s 2s/step - loss: 0.4421 - ac
c: 0.8209
51/51 [==============================] - 85s 2s/step - loss: 0.3880 - ac
c: 0.8484 - val_loss: 0.4421 - val_acc: 0.8209
Epoch 8/20
13/13 [==============================] - 28s 2s/step - loss: 0.4749 - ac
c: 0.8085
51/51 [==============================] - 85s 2s/step - loss: 0.3346 - ac
c: 0.8812 - val_loss: 0.4749 - val_acc: 0.8085
Epoch 9/20
13/13 [==============================] - 28s 2s/step - loss: 0.4725 - ac
```

```
c: 0.8109
51/51 [==============================] - 84s 2s/step - loss: 0.3398 - ac
c: 0.8719 - val_loss: 0.4725 - val_acc: 0.8109
Epoch 10/20
13/13 [==============================] - 28s 2s/step - loss: 0.4336 - ac
c: 0.8308
51/51 [==============================] - 86s 2s/step - loss: 0.3429 - ac
c: 0.8657 - val_loss: 0.4336 - val_acc: 0.8308
Epoch 11/20
13/13 [==============================] - 28s 2s/step - loss: 0.8778 - ac
c: 0.7164
51/51 [==============================] - 85s 2s/step - loss: 0.3140 - ac
c: 0.8793 - val_loss: 0.8778 - val_acc: 0.7164
Epoch 12/20
13/13 [==============================] - 27s 2s/step - loss: 0.4288 - ac
c: 0.8433
51/51 [==============================] - 86s 2s/step - loss: 0.3292 - ac
c: 0.8843 - val_loss: 0.4288 - val_acc: 0.8433
Epoch 13/20
13/13 [==============================] - 27s 2s/step - loss: 0.4722 - ac
c: 0.8259
51/51 [==============================] - 85s 2s/step - loss: 0.2988 - ac
c: 0.8837 - val_loss: 0.4722 - val_acc: 0.8259
Epoch 14/20
13/13 [==============================] - 28s 2s/step - loss: 0.4710 - ac
c: 0.8408
51/51 [==============================] - 85s 2s/step - loss: 0.2908 - ac
c: 0.8868 - val_loss: 0.4710 - val_acc: 0.8408
Epoch 15/20
13/13 [==============================] - 28s 2s/step - loss: 0.4941 - ac
c: 0.8308
51/51 [==============================] - 84s 2s/step - loss: 0.3021 - ac
c: 0.8818 - val_loss: 0.4941 - val_acc: 0.8308
Epoch 16/20
13/13 [==============================] - 27s 2s/step - loss: 0.6483 - ac
c: 0.8060
51/51 [==============================] - 83s 2s/step - loss: 0.2574 - ac
c: 0.9004 - val_loss: 0.6483 - val_acc: 0.8060
Epoch 17/20
13/13 [==============================] - 27s 2s/step - loss: 0.4420 - ac
c: 0.8333
51/51 [==============================] - 86s 2s/step - loss: 0.2927 - ac
c: 0.8923 - val_loss: 0.4420 - val_acc: 0.8333
Epoch 18/20
13/13 [==============================] - 28s 2s/step - loss: 0.4617 - ac
c: 0.8433
51/51 [==============================] - 85s 2s/step - loss: 0.2375 - ac
c: 0.9134 - val_loss: 0.4617 - val_acc: 0.8433
Epoch 19/20
13/13 [==============================] - 27s 2s/step - loss: 0.5188 - ac
c: 0.8134
51/51 [==============================] - 83s 2s/step - loss: 0.2808 - ac
c: 0.8861 - val_loss: 0.5188 - val_acc: 0.8134
Epoch 20/20
13/13 [==============================] - 28s 2s/step - loss: 0.4446 - ac
c: 0.8483
```

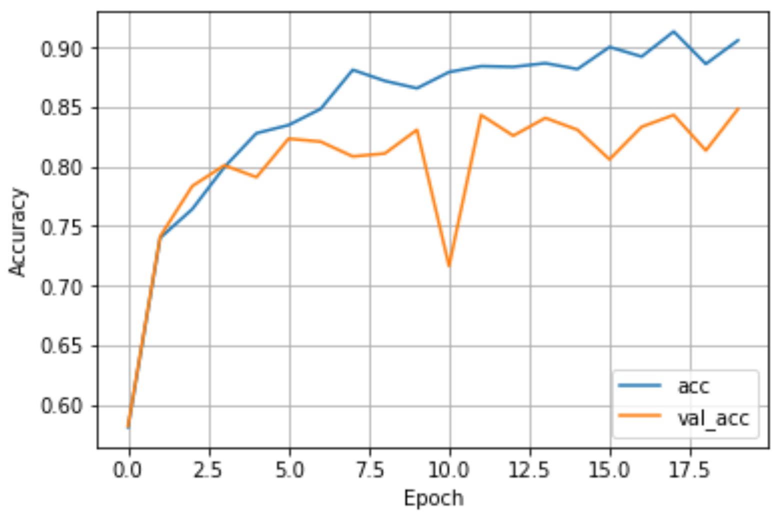
```
51/51 [==============================] - 84s 2s/step - loss: 0.2513 - ac
c: 0.9059 - val_loss: 0.4446 - val_acc: 0.8483
```

In [10]:

```
hist_his = hist.history
acc = hist_his['acc']
val_acc = hist_his['val_acc']
plt.plot(acc)
plt.plot(val_acc)
plt.grid()
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['acc','val_acc'], loc = 4)
```

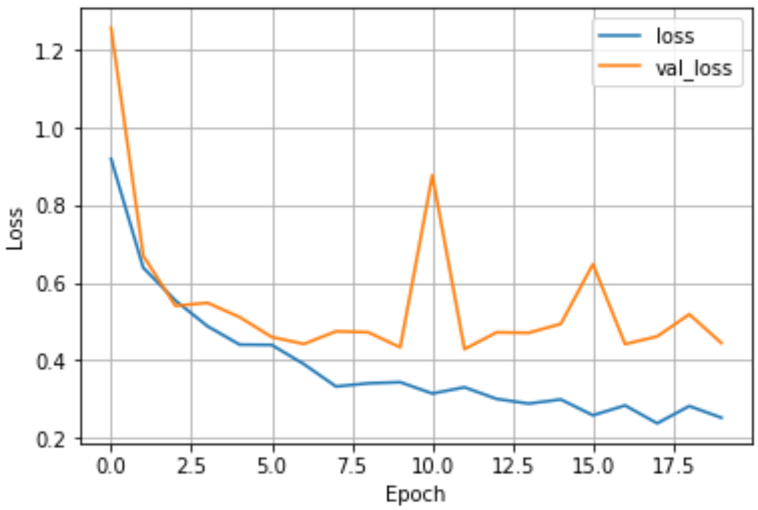
Out[10]:

```
<matplotlib.legend.Legend at 0x7ff13c577828>
```



In [11]:
```
loss = hist_his['loss']
val_loss = hist_his['val_loss']
plt.plot(loss)
plt.plot(val_loss)
plt.grid()
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['loss','val_loss'], loc = 1)
```

Out[11]: <matplotlib.legend.Legend at 0x7ff13c481518>



In [12]:
```
print(loss)
print(val_loss)
print(acc)
print(val_acc)
```

```
[0.9197668498105341, 0.6389658262233923, 0.5544988536598658, 0.4881614077
799391, 0.44060463657473575, 0.43979209454933016, 0.3903113628673081, 0.3
3251514175150654, 0.34068196362788133, 0.3435912899451681, 0.314347134663
3269, 0.3304667539230668, 0.3004123606894276, 0.2883142184207935, 0.29904
315701805717, 0.2581148448556957, 0.283596468001309, 0.23742855900880133,
0.28195172607308566, 0.25184559349966523]
[1.2577795065366304, 0.6699514022240272, 0.5406953211014087, 0.5484803112
653586, 0.5115789484519225, 0.4601581853169661, 0.44214536593510556, 0.47
486457228660583, 0.4725213853212503, 0.43356338372597325, 0.8777518776746
897, 0.42876676240792644, 0.47224560150733363, 0.47098710445257336, 0.494
0937597018022, 0.6483332835710965, 0.44195979833602905, 0.461708098649978
64, 0.5188222928689077, 0.444645609993201]
[0.58106434, 0.740099, 0.7642327, 0.79950494, 0.8279703, 0.83477724, 0.84
83911, 0.8811881, 0.8719059, 0.8657178, 0.8793317, 0.8842822, 0.88366336,
0.88675743, 0.8818069, 0.9003713, 0.8923267, 0.9133663, 0.8861386, 0.9059
406]
[0.58208954, 0.74129355, 0.7835821, 0.80099505, 0.7910448, 0.8233831, 0.8
208955, 0.80845773, 0.8109453, 0.8308458, 0.7164179, 0.8432836, 0.8258706
3, 0.840796, 0.8308458, 0.80597013, 0.8333333, 0.8432836, 0.8134328, 0.84
825873]
```

In [ ]:

Run VGG16 on 3 classes test2

For this part, we still loads 3 classes with most paintings in VGG16 while adds two more layers than test1. Still, run 20 epochs.

Before adding layers, we let the `base_model.output` load into variable `x`. Then, we just operate on the `x`. The operations are as following:

- A `Flatten()(x)` layer which reshapes the outputs to a single channel.
- (new) A `GaussianNoise(0.1)(x)` layer.
- (new) A `Dropout(0.5)(x)` layer.
- A fully-connected layer with 2304 output units and `relu` activation.
- A `GaussianNoise(0.1)(x)` layer.
- A `Dropout(0.5)(x)` layer.
- A fully-connected layer with 288 output units and `relu` activation.
- A `BatchNormalization()(x)` layer.
- A `Dropout(0.5)(x)` layer.
- A final fully-connected layer. Since this is a multiple classification, there should be three output and `softmax` activation. To mitigate overfitting, we add several arguments: `kernel_initializer='random_uniform'`, `bias_initializer='random_uniform'`, and `bias_regularizer=regularizers.l2(0.01)`.

Fortunately, at the end of this test2, we got nicer loss curves which reflects no obvious overfitting.

In [1]:
```
import numpy as np
import os
from tensorflow.keras import applications
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import optimizers, regularizers
from tensorflow.keras.models import Sequential
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dropout, Flatten, Dense, GaussianNoise,
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())
```

```
[name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality {
}
incarnation: 16256100486386227842
, name: "/device:GPU:0"
device_type: "GPU"
memory_limit: 2264907776
locality {
  bus_id: 1
  links {
  }
}
incarnation: 2620730678667875277
physical_device_desc: "device: 0, name: GeForce GTX 970M, pci bus id: 000
0:01:00.0, compute capability: 5.2"
]
```

In [3]:
```
import tensorflow.keras.backend as K
K.clear_session()
```

In [3]:
```
nrow = 200
ncol = 200
nclass = 3
base_model = applications.VGG16(weights='imagenet', input_shape=(nrow,ncol,3
for layer in base_model.layers:
    layer.trainable = False

x = base_model.output
x = Flatten()(x)
x = GaussianNoise(0.1)(x)
x = Dropout(0.5)(x)
x = Dense(2304, activation = 'relu')(x) # 18432/4
x = GaussianNoise(0.1)(x) # add noise to mitigate overfitting (regularizatio
x = Dropout(0.5)(x)
x = Dense(288, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)
pred = Dense(nclass, activation='softmax',
             kernel_initializer='random_uniform',
             bias_initializer='random_uniform',
             bias_regularizer=regularizers.l2(0.01),
             name='predictions')(x)
model = Model(inputs=base_model.input, outputs=pred)
```

```
WARNING:tensorflow:From D:\Anaconda\envs\tensorflow\lib\site-packages\ten
sorflow\python\ops\resource_variable_ops.py:435: colocate_with (from tens
orflow.python.framework.ops) is deprecated and will be removed in a futur
e version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From D:\Anaconda\envs\tensorflow\lib\site-packages\ten
sorflow\python\keras\layers\core.py:143: calling dropout (from tensorflo
w.python.ops.nn_ops) with keep_prob is deprecated and will be removed in
a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1
- keep_prob`.
```

In [4]:
```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         (None, 200, 200, 3)       0         
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 200, 200, 64)      1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 200, 200, 64)      36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 100, 100, 64)      0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 100, 100, 128)     73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 100, 100, 128)     147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 50, 50, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 50, 50, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 50, 50, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 50, 50, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 25, 25, 256)       0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 25, 25, 512)       1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 25, 25, 512)       2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 25, 25, 512)       2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 12, 12, 512)       0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 12, 12, 512)       2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 12, 12, 512)       2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 12, 12, 512)       2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 6, 6, 512)         0         
_________________________________________________________________
flatten (Flatten)            (None, 18432)             0         
_________________________________________________________________
gaussian_noise (GaussianNois (None, 18432)             0         
_________________________________________________________________
dropout (Dropout)            (None, 18432)             0         
_________________________________________________________________
dense (Dense)                (None, 2304)              42469632  
_________________________________________________________________
gaussian_noise_1 (GaussianNo (None, 2304)              0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 2304)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 288)               663840    
_________________________________________________________________
```

```
batch_normalization_v1 (Batc (None, 288)               1152
_________________________________________________________________
dropout_2 (Dropout)          (None, 288)               0
_________________________________________________________________
predictions (Dense)          (None, 3)                 867
=================================================================
Total params: 57,850,179
Trainable params: 43,134,915
Non-trainable params: 14,715,264
_________________________________________________________________
```

In [5]:

```
train_data_dir = './images_train'
batch_size = 10
train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)
train_generator = train_datagen.flow_from_directory(
                        train_data_dir,
                        target_size=(nrow,ncol),
                        batch_size=batch_size,
                        class_mode='categorical')
```

```
Found 1616 images belonging to 3 classes.
```

In [6]:

```
test_data_dir = './images_test'
batch_size = 10
test_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)
test_generator = train_datagen.flow_from_directory(
                        test_data_dir,
                        target_size=(nrow,ncol),
                        batch_size=batch_size,
                        class_mode='categorical')
```

```
Found 402 images belonging to 3 classes.
```

In [7]:

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['a
steps_per_epoch = train_generator.n // batch_size
validation_steps = test_generator.n // batch_size
```

In [8]:
```
nepochs = 20  # Number of epochs

# Call the fit_generator function
hist = model.fit_generator(
    train_generator,
    epochs=nepochs,
    steps_per_epoch=steps_per_epoch,
    validation_data=test_generator,
    validation_steps=validation_steps)
```

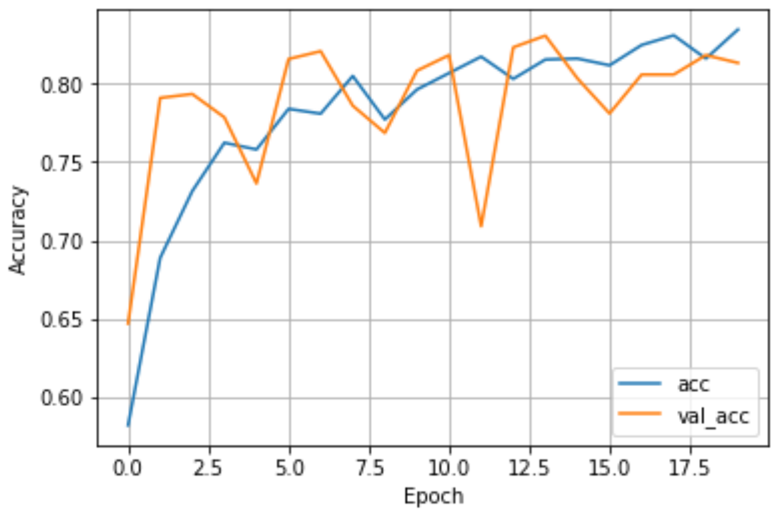
```
WARNING:tensorflow:From D:\Anaconda\envs\tensorflow\lib\site-packages\ten
sorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.python.op
s.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/20
41/41 [==============================] - 11s 269ms/step - loss: 0.8014 -
acc: 0.6468
162/162 [==============================] - 51s 314ms/step - loss: 0.9062
- acc: 0.5817 - val_loss: 0.8014 - val_acc: 0.6468
Epoch 2/20
41/41 [==============================] - 10s 234ms/step - loss: 0.5261 -
acc: 0.7910
162/162 [==============================] - 48s 299ms/step - loss: 0.7397
- acc: 0.6887 - val_loss: 0.5261 - val_acc: 0.7910
Epoch 3/20
41/41 [==============================] - 11s 272ms/step - loss: 0.4968 -
acc: 0.7935
162/162 [==============================] - 48s 294ms/step - loss: 0.6464
- acc: 0.7314 - val_loss: 0.4968 - val_acc: 0.7935
Epoch 4/20
41/41 [==============================] - 9s 230ms/step - loss: 0.5469 - a
cc: 0.7786
162/162 [==============================] - 48s 298ms/step - loss: 0.6067
- acc: 0.7624 - val_loss: 0.5469 - val_acc: 0.7786
Epoch 5/20
41/41 [==============================] - 10s 242ms/step - loss: 0.6292 -
acc: 0.7363
162/162 [==============================] - 47s 290ms/step - loss: 0.5992
- acc: 0.7580 - val_loss: 0.6292 - val_acc: 0.7363
Epoch 6/20
41/41 [==============================] - 9s 224ms/step - loss: 0.4780 - a
cc: 0.8159
162/162 [==============================] - 46s 286ms/step - loss: 0.5729
- acc: 0.7840 - val_loss: 0.4780 - val_acc: 0.8159
Epoch 7/20
41/41 [==============================] - 9s 220ms/step - loss: 0.4713 - a
cc: 0.8209
162/162 [==============================] - 46s 287ms/step - loss: 0.5742
- acc: 0.7809 - val_loss: 0.4713 - val_acc: 0.8209
Epoch 8/20
41/41 [==============================] - 10s 254ms/step - loss: 0.5374 -
acc: 0.7861
162/162 [==============================] - 48s 297ms/step - loss: 0.5082
- acc: 0.8051 - val_loss: 0.5374 - val_acc: 0.7861
Epoch 9/20
41/41 [==============================] - 9s 222ms/step - loss: 0.5552 - a
```

```
cc: 0.7687
162/162 [==============================] - 46s 286ms/step - loss: 0.5573
- acc: 0.7772 - val_loss: 0.5552 - val_acc: 0.7687
Epoch 10/20
41/41 [==============================] - 10s 252ms/step - loss: 0.4952 -
acc: 0.8085
162/162 [==============================] - 48s 294ms/step - loss: 0.5131
- acc: 0.7964 - val_loss: 0.4952 - val_acc: 0.8085
Epoch 11/20
41/41 [==============================] - 9s 222ms/step - loss: 0.4551 - a
cc: 0.8184
162/162 [==============================] - 46s 287ms/step - loss: 0.4911
- acc: 0.8069 - val_loss: 0.4551 - val_acc: 0.8184
Epoch 12/20
41/41 [==============================] - 10s 246ms/step - loss: 0.7736 -
acc: 0.7090
162/162 [==============================] - 48s 298ms/step - loss: 0.4689
- acc: 0.8175 - val_loss: 0.7736 - val_acc: 0.7090
Epoch 13/20
41/41 [==============================] - 9s 223ms/step - loss: 0.4404 - a
cc: 0.8234
162/162 [==============================] - 46s 282ms/step - loss: 0.4993
- acc: 0.8032 - val_loss: 0.4404 - val_acc: 0.8234
Epoch 14/20
41/41 [==============================] - 11s 269ms/step - loss: 0.4481 -
acc: 0.8308
162/162 [==============================] - 48s 299ms/step - loss: 0.4822
- acc: 0.8156 - val_loss: 0.4481 - val_acc: 0.8308
Epoch 15/20
41/41 [==============================] - 9s 220ms/step - loss: 0.5458 - a
cc: 0.8035
162/162 [==============================] - 46s 284ms/step - loss: 0.4653
- acc: 0.8162 - val_loss: 0.5458 - val_acc: 0.8035
Epoch 16/20
41/41 [==============================] - 9s 232ms/step - loss: 0.5927 - a
cc: 0.7811
162/162 [==============================] - 47s 290ms/step - loss: 0.4841
- acc: 0.8119 - val_loss: 0.5927 - val_acc: 0.7811
Epoch 17/20
41/41 [==============================] - 10s 254ms/step - loss: 0.4682 -
acc: 0.8060
162/162 [==============================] - 47s 293ms/step - loss: 0.4577
- acc: 0.8249 - val_loss: 0.4682 - val_acc: 0.8060
Epoch 18/20
41/41 [==============================] - 10s 241ms/step - loss: 0.4739 -
acc: 0.8060
162/162 [==============================] - 47s 293ms/step - loss: 0.4303
- acc: 0.8311 - val_loss: 0.4739 - val_acc: 0.8060
Epoch 19/20
41/41 [==============================] - 10s 237ms/step - loss: 0.4896 -
acc: 0.8184
162/162 [==============================] - 49s 302ms/step - loss: 0.4780
- acc: 0.8162 - val_loss: 0.4896 - val_acc: 0.8184
Epoch 20/20
41/41 [==============================] - 11s 261ms/step - loss: 0.4374 -
acc: 0.8134
```

```
162/162 [==============================] - 51s 315ms/step - loss: 0.4418
- acc: 0.8348 - val_loss: 0.4374 - val_acc: 0.8134
```

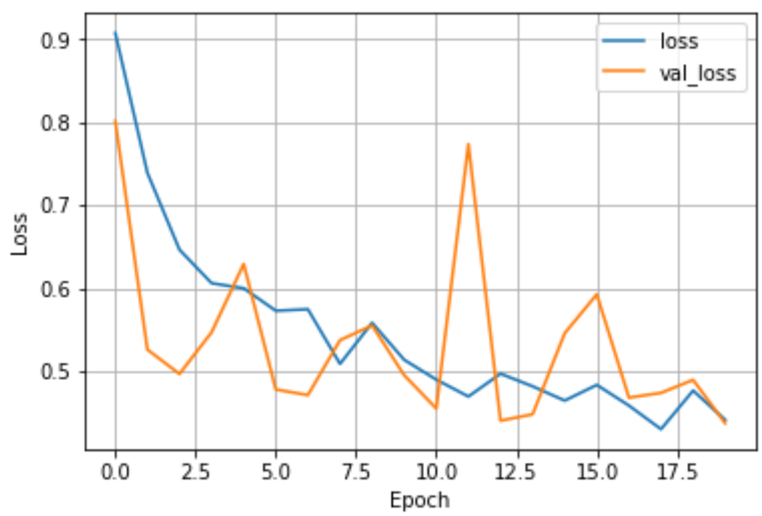
In [9]:
```
hist_his = hist.history
acc = hist_his['acc']
val_acc = hist_his['val_acc']
plt.plot(acc)
plt.plot(val_acc)
plt.grid()
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['acc','val_acc'], loc = 4)
```

Out[9]:
```
<matplotlib.legend.Legend at 0x1b6acc61898>
```



In [10]:
```
loss = hist_his['loss']
val_loss = hist_his['val_loss']
plt.plot(loss)
plt.plot(val_loss)
plt.grid()
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['loss','val_loss'], loc = 1)
```

Out[10]: <matplotlib.legend.Legend at 0x1b6c6575f98>



In [11]:
```
print(loss)
print(val_loss)
print(acc)
print(val_acc)
```

```
[0.9074013190030461, 0.7389471186970425, 0.6464045981295629, 0.6060428023
246107, 0.5999550910131766, 0.5728324980539555, 0.5746590845047099, 0.508
8578708617404, 0.558343212138677, 0.5137309753658748, 0.4898616761743727,
0.4695385329628197, 0.4970195085861453, 0.48181152163018093, 0.4646304168
1575007, 0.483507089943874, 0.4586440159911566, 0.43021603463457364, 0.47
66368215331937, 0.4413660408670802]
[0.8013866336607351, 0.5260722466358324, 0.49675964255158495, 0.546905299
8757944, 0.6292213034339067, 0.47799311978061026, 0.471272920599071, 0.53
74162589631429, 0.5552048854893301, 0.49515536618305417, 0.45510127018319
396, 0.7736208001833137, 0.44041226286350227, 0.4480742525036742, 0.54579
53320043843, 0.5926764856387929, 0.468191360918487, 0.4738691456434203,
0.48956028026778525, 0.43735222977290794]
[0.58168316, 0.68873763, 0.73143566, 0.76237625, 0.75804454, 0.78403467,
0.7809406, 0.8050743, 0.7772277, 0.7964109, 0.8069307, 0.8174505, 0.80321
78, 0.8155941, 0.8162129, 0.8118812, 0.82487625, 0.83106434, 0.8162129,
0.83477724]
[0.6467662, 0.7910448, 0.7935323, 0.77860695, 0.7363184, 0.8159204, 0.820
8955, 0.78606963, 0.76865673, 0.80845773, 0.81840795, 0.7089552, 0.823383
1, 0.8308458, 0.8034826, 0.78109455, 0.80597013, 0.80597013, 0.81840795,
0.8134328]
```

In [ ]:

Run VGG16 on 3 classes test3

For this part, we still loads 3 classes with most paintings in VGG16 while adds two more layers than test2. Still, run 20 epochs.

Before adding layers, we let the `base_model.output` load into variable `x`. Then, we just operate on the `x`. The operations are as following:

- A `Flatten()(x)` layer which reshapes the outputs to a single channel.
- A `GaussianNoise(0.1)(x)` layer.
- A `Dropout(0.5)(x)` layer.
- A fully-connected layer with 2304 output units and `relu` activation.
- A `GaussianNoise(0.1)(x)` layer.
- A `Dropout(0.5)(x)` layer.
- A fully-connected layer with 288 output units and `relu` activation.
- A `BatchNormalization()(x)` layer.
- A `Dropout(0.5)(x)` layer.
- (new) A fully-connected layer with 288 output units and `relu` activation.
- (new) A `Dropout(0.5)(x)` layer.
- A final fully-connected layer. Since this is a multiple classification, there should be three output and `softmax` activation. To mitigate overfitting, we add several arguments: `kernel_initializer='random_uniform'`, `bias_initializer='random_uniform'`, and `bias_regularizer=regularizers.l2(0.01)`.

However, at the end of test3, we end up to an obvious underfitting.

In [1]:
```
import numpy as np
import os
from tensorflow.keras import applications
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import optimizers, regularizers
from tensorflow.keras.models import Sequential
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dropout, Flatten, Dense, GaussianNoise,
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())
```

```
[name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality {
}
incarnation: 2638700649349130033
, name: "/device:GPU:0"
device_type: "GPU"
memory_limit: 2264907776
locality {
  bus_id: 1
  links {
  }
}
incarnation: 14776539995965970595
physical_device_desc: "device: 0, name: GeForce GTX 970M, pci bus id: 000
0:01:00.0, compute capability: 5.2"
]
```

In [3]:
```
import tensorflow.keras.backend as K
K.clear_session()
```

In [4]:
```
nrow = 200
ncol = 200
nclass = 3
base_model = applications.VGG16(weights='imagenet', input_shape=(nrow,ncol,3
for layer in base_model.layers:
    layer.trainable = False

x = base_model.output
x = Flatten()(x)
x = GaussianNoise(0.1)(x)
x = Dropout(0.5)(x)
x = Dense(2304, activation = 'relu')(x) # 18432/4
x = GaussianNoise(0.1)(x) # add noise to mitigate overfitting (regularizatic
x = Dropout(0.5)(x)
x = Dense(288, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)
x = Dense(288, activation='relu')(x)
x = Dropout(0.5)(x)
pred = Dense(nclass, activation='softmax',
             kernel_initializer='random_uniform',
             bias_initializer='random_uniform',
             bias_regularizer=regularizers.l2(0.01),
             name='predictions')(x)
model = Model(inputs=base_model.input, outputs=pred)
```

```
WARNING:tensorflow:From D:\Anaconda\envs\tensorflow\lib\site-packages\ten
sorflow\python\ops\resource_variable_ops.py:435: colocate_with (from tens
orflow.python.framework.ops) is deprecated and will be removed in a futur
e version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From D:\Anaconda\envs\tensorflow\lib\site-packages\ten
sorflow\python\keras\layers\core.py:143: calling dropout (from tensorflo
w.python.ops.nn_ops) with keep_prob is deprecated and will be removed in
a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1
- keep_prob`.
```

In [5]:
```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         (None, 200, 200, 3)       0         
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 200, 200, 64)      1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 200, 200, 64)      36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 100, 100, 64)      0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 100, 100, 128)     73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 100, 100, 128)     147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 50, 50, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 50, 50, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 50, 50, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 50, 50, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 25, 25, 256)       0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 25, 25, 512)       1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 25, 25, 512)       2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 25, 25, 512)       2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 12, 12, 512)       0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 12, 12, 512)       2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 12, 12, 512)       2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 12, 12, 512)       2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 6, 6, 512)         0         
_________________________________________________________________
flatten (Flatten)            (None, 18432)             0         
_________________________________________________________________
gaussian_noise (GaussianNois (None, 18432)             0         
_________________________________________________________________
dropout (Dropout)            (None, 18432)             0         
_________________________________________________________________
dense (Dense)                (None, 2304)              42469632  
_________________________________________________________________
gaussian_noise_1 (GaussianNo (None, 2304)              0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 2304)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 288)               663840    
_________________________________________________________________
```

```
batch_normalization_v1 (Batc (None, 288)               1152
_________________________________________________________________
dropout_2 (Dropout)          (None, 288)               0
_________________________________________________________________
dense_2 (Dense)              (None, 288)               83232
_________________________________________________________________
dropout_3 (Dropout)          (None, 288)               0
_________________________________________________________________
predictions (Dense)          (None, 3)                 867
=================================================================
Total params: 57,933,411
Trainable params: 43,218,147
Non-trainable params: 14,715,264
_________________________________________________________________
```

In [6]:

```
train_data_dir = './images_train'
batch_size = 5
train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)
train_generator = train_datagen.flow_from_directory(
                        train_data_dir,
                        target_size=(nrow,ncol),
                        batch_size=batch_size,
                        class_mode='categorical')
```

```
Found 1616 images belonging to 3 classes.
```

In [7]:

```
test_data_dir = './images_test'
batch_size = 5
test_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)
test_generator = train_datagen.flow_from_directory(
                        test_data_dir,
                        target_size=(nrow,ncol),
                        batch_size=batch_size,
                        class_mode='categorical')
```

```
Found 402 images belonging to 3 classes.
```

In [8]:

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['a
steps_per_epoch = train_generator.n // batch_size
validation_steps = test_generator.n // batch_size
```

In [9]:
```
nepochs = 20  # Number of epochs

# Call the fit_generator function
hist = model.fit_generator(
    train_generator,
    epochs=nepochs,
    steps_per_epoch=steps_per_epoch,
    validation_data=test_generator,
    validation_steps=validation_steps)
```

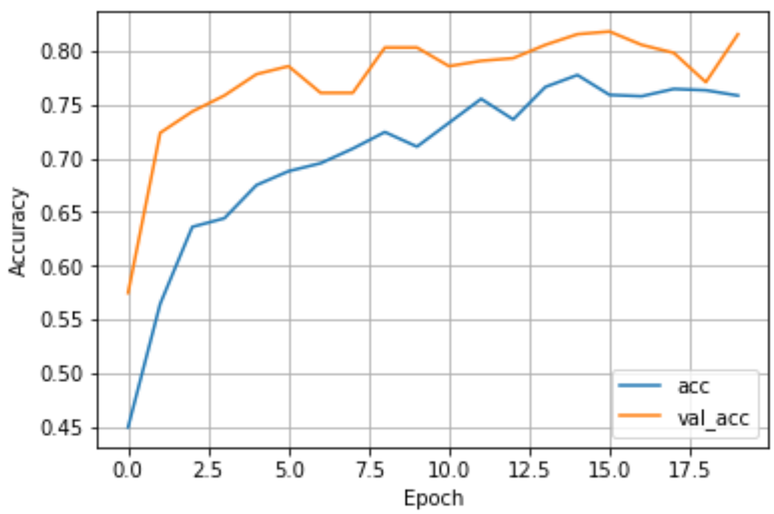
```
WARNING:tensorflow:From D:\Anaconda\envs\tensorflow\lib\site-packages\ten
sorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.python.op
s.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/20
81/81 [==============================] - 11s 137ms/step - loss: 0.8665 -
acc: 0.5746
324/324 [==============================] - 58s 178ms/step - loss: 1.0749
- acc: 0.4493 - val_loss: 0.8665 - val_acc: 0.5746
Epoch 2/20
81/81 [==============================] - 10s 126ms/step - loss: 0.6374 -
acc: 0.7239
324/324 [==============================] - 56s 172ms/step - loss: 0.9348
- acc: 0.5644 - val_loss: 0.6374 - val_acc: 0.7239
Epoch 3/20
81/81 [==============================] - 10s 126ms/step - loss: 0.6223 -
acc: 0.7438
324/324 [==============================] - 55s 169ms/step - loss: 0.8278
- acc: 0.6361 - val_loss: 0.6223 - val_acc: 0.7438
Epoch 4/20
81/81 [==============================] - 11s 136ms/step - loss: 0.6065 -
acc: 0.7587
324/324 [==============================] - 56s 173ms/step - loss: 0.8256
- acc: 0.6442 - val_loss: 0.6065 - val_acc: 0.7587
Epoch 5/20
81/81 [==============================] - 9s 116ms/step - loss: 0.5289 - a
cc: 0.7786
324/324 [==============================] - 55s 170ms/step - loss: 0.7670
- acc: 0.6751 - val_loss: 0.5289 - val_acc: 0.7786
Epoch 6/20
81/81 [==============================] - 11s 135ms/step - loss: 0.5223 -
acc: 0.7861
324/324 [==============================] - 56s 173ms/step - loss: 0.7464
- acc: 0.6881 - val_loss: 0.5223 - val_acc: 0.7861
Epoch 7/20
81/81 [==============================] - 11s 138ms/step - loss: 0.5701 -
acc: 0.7612
324/324 [==============================] - 56s 172ms/step - loss: 0.7299
- acc: 0.6955 - val_loss: 0.5701 - val_acc: 0.7612
Epoch 8/20
81/81 [==============================] - 10s 120ms/step - loss: 0.5727 -
acc: 0.7612
324/324 [==============================] - 54s 166ms/step - loss: 0.7158
- acc: 0.7092 - val_loss: 0.5727 - val_acc: 0.7612
Epoch 9/20
81/81 [==============================] - 11s 132ms/step - loss: 0.5174 -
```

```
acc: 0.8035
324/324 [==============================] - 56s 173ms/step - loss: 0.6799
- acc: 0.7246 - val_loss: 0.5174 - val_acc: 0.8035
Epoch 10/20
81/81 [==============================] - 10s 119ms/step - loss: 0.4813 -
acc: 0.8035
324/324 [==============================] - 54s 165ms/step - loss: 0.6832
- acc: 0.7110 - val_loss: 0.4813 - val_acc: 0.8035
Epoch 11/20
81/81 [==============================] - 11s 134ms/step - loss: 0.5021 -
acc: 0.7861
324/324 [==============================] - 57s 176ms/step - loss: 0.6640
- acc: 0.7333 - val_loss: 0.5021 - val_acc: 0.7861
Epoch 12/20
81/81 [==============================] - 10s 121ms/step - loss: 0.5168 -
acc: 0.7910
324/324 [==============================] - 54s 165ms/step - loss: 0.6337
- acc: 0.7556 - val_loss: 0.5168 - val_acc: 0.7910
Epoch 13/20
81/81 [==============================] - 10s 129ms/step - loss: 0.5120 -
acc: 0.7935
324/324 [==============================] - 56s 173ms/step - loss: 0.6604
- acc: 0.7364 - val_loss: 0.5120 - val_acc: 0.7935
Epoch 14/20
81/81 [==============================] - 10s 119ms/step - loss: 0.5025 -
acc: 0.8060
324/324 [==============================] - 54s 168ms/step - loss: 0.6215
- acc: 0.7667 - val_loss: 0.5025 - val_acc: 0.8060
Epoch 15/20
81/81 [==============================] - 10s 123ms/step - loss: 0.4450 -
acc: 0.8159
324/324 [==============================] - 55s 171ms/step - loss: 0.5811
- acc: 0.7778 - val_loss: 0.4450 - val_acc: 0.8159
Epoch 16/20
81/81 [==============================] - 11s 135ms/step - loss: 0.4971 -
acc: 0.8184
324/324 [==============================] - 55s 169ms/step - loss: 0.6520
- acc: 0.7593 - val_loss: 0.4971 - val_acc: 0.8184
Epoch 17/20
81/81 [==============================] - 10s 129ms/step - loss: 0.5052 -
acc: 0.8060
324/324 [==============================] - 55s 171ms/step - loss: 0.6010
- acc: 0.7580 - val_loss: 0.5052 - val_acc: 0.8060
Epoch 18/20
81/81 [==============================] - 10s 124ms/step - loss: 0.5317 -
acc: 0.7985
324/324 [==============================] - 54s 167ms/step - loss: 0.6009
- acc: 0.7649 - val_loss: 0.5317 - val_acc: 0.7985
Epoch 19/20
81/81 [==============================] - 10s 124ms/step - loss: 0.5673 -
acc: 0.7711
324/324 [==============================] - 55s 170ms/step - loss: 0.6075
- acc: 0.7636 - val_loss: 0.5673 - val_acc: 0.7711
Epoch 20/20
81/81 [==============================] - 11s 136ms/step - loss: 0.4825 -
acc: 0.8159
```

```
324/324 [==============================] - 55s 171ms/step - loss: 0.6141
- acc: 0.7587 - val_loss: 0.4825 - val_acc: 0.8159
```

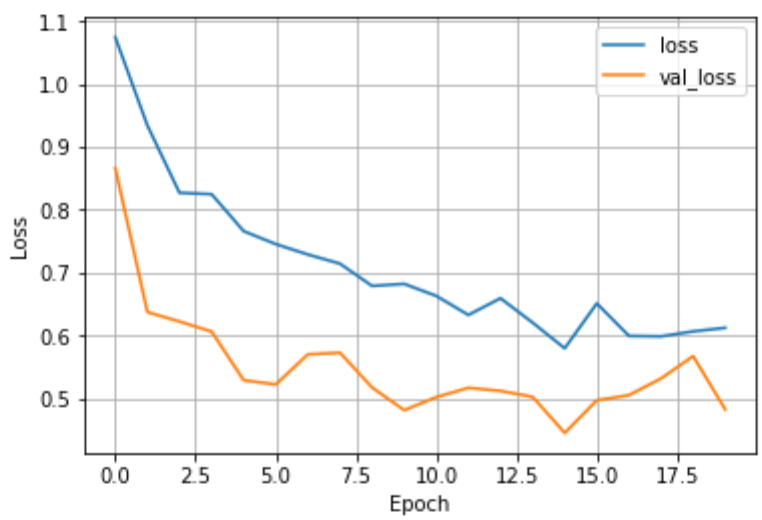
In [10]:
```
hist_his = hist.history
acc = hist_his['acc']
val_acc = hist_his['val_acc']
plt.plot(acc)
plt.plot(val_acc)
plt.grid()
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['acc','val_acc'], loc = 4)
```

Out[10]: <matplotlib.legend.Legend at 0x1d2c6043630>



In [11]:
```
loss = hist_his['loss']
val_loss = hist_his['val_loss']
plt.plot(loss)
plt.plot(val_loss)
plt.grid()
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['loss','val_loss'], loc = 1)
```

Out[11]: <matplotlib.legend.Legend at 0x1d2dfacfef0>



In [12]:
```
print(loss)
print(val_loss)
print(acc)
print(val_acc)
```

```
[1.0747226900651607, 0.9344037247286869, 0.8271783235953143, 0.8250294966
0146, 0.7661922124795394, 0.7455206539131479, 0.7290131777091561, 0.71442
07797557263, 0.6789925487131102, 0.6822495735204308, 0.6631154680191217,
0.6327936213860048, 0.6593814821070115, 0.6206221104424336, 0.57972806726
35643, 0.651208940951043, 0.5995261630119461, 0.5987160129813791, 0.60664
43066546084, 0.6123226203789731]
[0.8664734724127217, 0.6373515651549821, 0.6223491394409427, 0.6064664299
289385, 0.5288627569874128, 0.5222727093431685, 0.5701493821854209, 0.572
7167229777501, 0.5173582800744493, 0.48125729663872424, 0.502123031351301
4, 0.5168457585789356, 0.5119663116142705, 0.5025214562422515, 0.44499328
03112416, 0.49711152147731663, 0.5052004380174625, 0.5317020053503874, 0.
5672682631059469, 0.48254425101626064]
[0.44925743, 0.56435645, 0.6361386, 0.64418316, 0.67512375, 0.6881188, 0.
69554454, 0.7091584, 0.7246287, 0.71101487, 0.7332921, 0.7555693, 0.73638
61, 0.7667079, 0.7778465, 0.7592822, 0.75804454, 0.7648515, 0.7636139, 0.
75866336]
[0.57462686, 0.7238806, 0.7437811, 0.75870645, 0.77860695, 0.78606963, 0.
76119405, 0.76119405, 0.8034826, 0.8034826, 0.78606963, 0.7910448, 0.7935
323, 0.80597013, 0.8159204, 0.81840795, 0.80597013, 0.79850745, 0.771144
3, 0.8159204]
```

In [ ]:

First validation of test2

Based on all three tests' results, the test2 has the best performance. Thus, we decide to repeat the test2's model two more times which verify the modle may work well on 30 classes.

The results of this repeat goes well.

In [1]:
```
import numpy as np
import os
from tensorflow.keras import applications
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import optimizers, regularizers
from tensorflow.keras.models import Sequential
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dropout, Flatten, Dense, GaussianNoise,
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())
```

```
[name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality {
}
incarnation: 14699118327036049130
, name: "/device:GPU:0"
device_type: "GPU"
memory_limit: 2264907776
locality {
  bus_id: 1
  links {
  }
}
incarnation: 10900450880824840497
physical_device_desc: "device: 0, name: GeForce GTX 970M, pci bus id: 000
0:01:00.0, compute capability: 5.2"
]
```

In [3]:
```
import tensorflow.keras.backend as K
K.clear_session()
```

In [4]:
```
nrow = 200
ncol = 200
nclass = 3
base_model = applications.VGG16(weights='imagenet', input_shape=(nrow,ncol,3
for layer in base_model.layers:
    layer.trainable = False

x = base_model.output
x = Flatten()(x)
x = GaussianNoise(0.1)(x)
x = Dropout(0.5)(x)
x = Dense(2304, activation = 'relu')(x) # 18432/4
x = GaussianNoise(0.1)(x) # add noise to mitigate overfitting (regularizatio
x = Dropout(0.5)(x)
x = Dense(288, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)
pred = Dense(nclass, activation='softmax',
             kernel_initializer='random_uniform',
             bias_initializer='random_uniform',
             bias_regularizer=regularizers.l2(0.01),
             name='predictions')(x)
model = Model(inputs=base_model.input, outputs=pred)
```

```
WARNING:tensorflow:From D:\Anaconda\envs\tensorflow\lib\site-packages\ten
sorflow\python\ops\resource_variable_ops.py:435: colocate_with (from tens
orflow.python.framework.ops) is deprecated and will be removed in a futur
e version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From D:\Anaconda\envs\tensorflow\lib\site-packages\ten
sorflow\python\keras\layers\core.py:143: calling dropout (from tensorflo
w.python.ops.nn_ops) with keep_prob is deprecated and will be removed in
a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1
- keep_prob`.
```

In [5]:
```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         (None, 200, 200, 3)       0         
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 200, 200, 64)      1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 200, 200, 64)      36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 100, 100, 64)      0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 100, 100, 128)     73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 100, 100, 128)     147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 50, 50, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 50, 50, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 50, 50, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 50, 50, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 25, 25, 256)       0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 25, 25, 512)       1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 25, 25, 512)       2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 25, 25, 512)       2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 12, 12, 512)       0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 12, 12, 512)       2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 12, 12, 512)       2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 12, 12, 512)       2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 6, 6, 512)         0         
_________________________________________________________________
flatten (Flatten)            (None, 18432)             0         
_________________________________________________________________
gaussian_noise (GaussianNois (None, 18432)             0         
_________________________________________________________________
dropout (Dropout)            (None, 18432)             0         
_________________________________________________________________
dense (Dense)                (None, 2304)              42469632  
_________________________________________________________________
gaussian_noise_1 (GaussianNo (None, 2304)              0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 2304)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 288)               663840    
_________________________________________________________________
```

```
batch_normalization_v1 (Batc (None, 288)              1152
_________________________________________________________________
dropout_2 (Dropout)          (None, 288)              0
_________________________________________________________________
predictions (Dense)          (None, 3)                867
=================================================================
Total params: 57,850,179
Trainable params: 43,134,915
Non-trainable params: 14,715,264
_________________________________________________________________
```

In [6]:

```
train_data_dir = './images_train'
batch_size = 10
train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)
train_generator = train_datagen.flow_from_directory(
                        train_data_dir,
                        target_size=(nrow,ncol),
                        batch_size=batch_size,
                        class_mode='categorical')
```

```
Found 1616 images belonging to 3 classes.
```

In [7]:

```
test_data_dir = './images_test'
batch_size = 10
test_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)
test_generator = train_datagen.flow_from_directory(
                        test_data_dir,
                        target_size=(nrow,ncol),
                        batch_size=batch_size,
                        class_mode='categorical')
```

```
Found 402 images belonging to 3 classes.
```

In [8]:

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['a
steps_per_epoch = train_generator.n // batch_size
validation_steps = test_generator.n // batch_size
```

In [9]:
```
nepochs = 20  # Number of epochs

# Call the fit_generator function
hist = model.fit_generator(
    train_generator,
    epochs=nepochs,
    steps_per_epoch=steps_per_epoch,
    validation_data=test_generator,
    validation_steps=validation_steps)
```

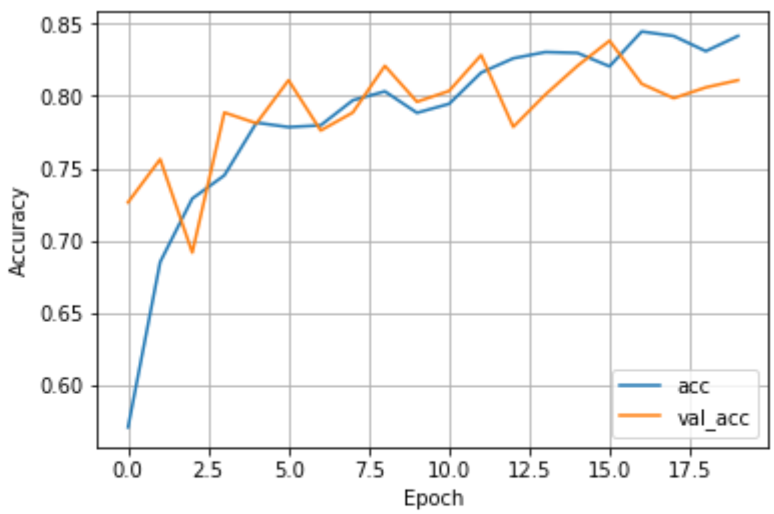
```
WARNING:tensorflow:From D:\Anaconda\envs\tensorflow\lib\site-packages\ten
sorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.python.op
s.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/20
41/41 [==============================] - 12s 281ms/step - loss: 0.7035 -
acc: 0.7264
162/162 [==============================] - 51s 316ms/step - loss: 0.9289
- acc: 0.5705 - val_loss: 0.7035 - val_acc: 0.7264
Epoch 2/20
41/41 [==============================] - 10s 234ms/step - loss: 0.5645 -
acc: 0.7562
162/162 [==============================] - 49s 303ms/step - loss: 0.7224
- acc: 0.6850 - val_loss: 0.5645 - val_acc: 0.7562
Epoch 3/20
41/41 [==============================] - 10s 251ms/step - loss: 0.7018 -
acc: 0.6915
162/162 [==============================] - 49s 300ms/step - loss: 0.6750
- acc: 0.7290 - val_loss: 0.7018 - val_acc: 0.6915
Epoch 4/20
41/41 [==============================] - 11s 262ms/step - loss: 0.5393 -
acc: 0.7886
162/162 [==============================] - 48s 299ms/step - loss: 0.6296
- acc: 0.7450 - val_loss: 0.5393 - val_acc: 0.7886
Epoch 5/20
41/41 [==============================] - 11s 268ms/step - loss: 0.5456 -
acc: 0.7811
162/162 [==============================] - 51s 315ms/step - loss: 0.5613
- acc: 0.7816 - val_loss: 0.5456 - val_acc: 0.7811
Epoch 6/20
41/41 [==============================] - 10s 238ms/step - loss: 0.4797 -
acc: 0.8109
162/162 [==============================] - 50s 306ms/step - loss: 0.5882
- acc: 0.7785 - val_loss: 0.4797 - val_acc: 0.8109
Epoch 7/20
41/41 [==============================] - 11s 272ms/step - loss: 0.5121 -
acc: 0.7761
162/162 [==============================] - 50s 309ms/step - loss: 0.5580
- acc: 0.7797 - val_loss: 0.5121 - val_acc: 0.7761
Epoch 8/20
41/41 [==============================] - 10s 234ms/step - loss: 0.6488 -
acc: 0.7886
162/162 [==============================] - 48s 295ms/step - loss: 0.5140
- acc: 0.7970 - val_loss: 0.6488 - val_acc: 0.7886
Epoch 9/20
41/41 [==============================] - 11s 271ms/step - loss: 0.4694 -
```

```
acc: 0.8209
162/162 [==============================] - 51s 313ms/step - loss: 0.5074
- acc: 0.8032 - val_loss: 0.4694 - val_acc: 0.8209
Epoch 10/20
41/41 [==============================] - 9s 232ms/step - loss: 0.4876 - a
cc: 0.7960
162/162 [==============================] - 49s 305ms/step - loss: 0.5211
- acc: 0.7884 - val_loss: 0.4876 - val_acc: 0.7960
Epoch 11/20
41/41 [==============================] - 9s 230ms/step - loss: 0.4746 - a
cc: 0.8035
162/162 [==============================] - 51s 312ms/step - loss: 0.5444
- acc: 0.7946 - val_loss: 0.4746 - val_acc: 0.8035
Epoch 12/20
41/41 [==============================] - 12s 297ms/step - loss: 0.4461 -
acc: 0.8284
162/162 [==============================] - 52s 321ms/step - loss: 0.4821
- acc: 0.8162 - val_loss: 0.4461 - val_acc: 0.8284
Epoch 13/20
41/41 [==============================] - 11s 262ms/step - loss: 0.5734 -
acc: 0.7786
162/162 [==============================] - 56s 346ms/step - loss: 0.4607
- acc: 0.8261 - val_loss: 0.5734 - val_acc: 0.7786
Epoch 14/20
41/41 [==============================] - 10s 236ms/step - loss: 0.4948 -
acc: 0.8010
162/162 [==============================] - 49s 302ms/step - loss: 0.4501
- acc: 0.8304 - val_loss: 0.4948 - val_acc: 0.8010
Epoch 15/20
41/41 [==============================] - 10s 249ms/step - loss: 0.4757 -
acc: 0.8209
162/162 [==============================] - 49s 303ms/step - loss: 0.4397
- acc: 0.8298 - val_loss: 0.4757 - val_acc: 0.8209
Epoch 16/20
41/41 [==============================] - 10s 253ms/step - loss: 0.4391 -
acc: 0.8383
162/162 [==============================] - 50s 309ms/step - loss: 0.4537
- acc: 0.8205 - val_loss: 0.4391 - val_acc: 0.8383
Epoch 17/20
41/41 [==============================] - 12s 289ms/step - loss: 0.4942 -
acc: 0.8085
162/162 [==============================] - 50s 308ms/step - loss: 0.4363
- acc: 0.8447 - val_loss: 0.4942 - val_acc: 0.8085
Epoch 18/20
41/41 [==============================] - 10s 232ms/step - loss: 0.5018 -
acc: 0.7985
162/162 [==============================] - 49s 303ms/step - loss: 0.4435
- acc: 0.8416 - val_loss: 0.5018 - val_acc: 0.7985
Epoch 19/20
41/41 [==============================] - 11s 261ms/step - loss: 0.5036 -
acc: 0.8060
162/162 [==============================] - 50s 310ms/step - loss: 0.4563
- acc: 0.8311 - val_loss: 0.5036 - val_acc: 0.8060
Epoch 20/20
41/41 [==============================] - 11s 259ms/step - loss: 0.4577 -
acc: 0.8109
```

```
162/162 [==============================] - 48s 299ms/step - loss: 0.4124
- acc: 0.8416 - val_loss: 0.4577 - val_acc: 0.8109
```

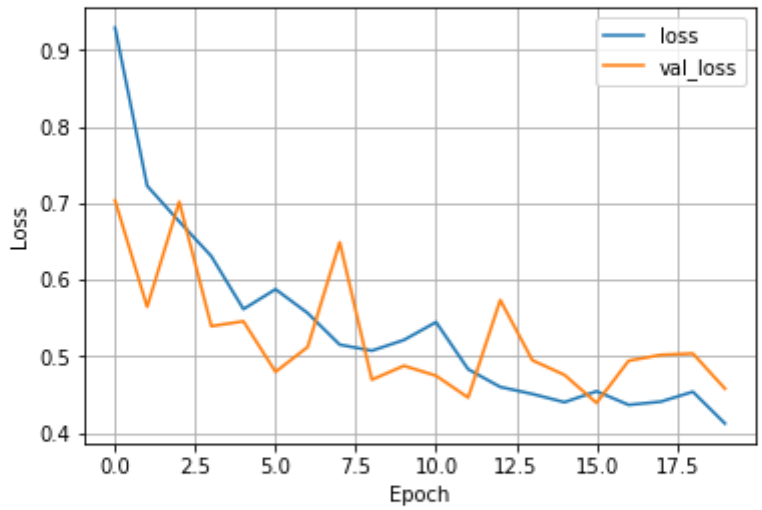
In [10]:
```
hist_his = hist.history
acc = hist_his['acc']
val_acc = hist_his['val_acc']
plt.plot(acc)
plt.plot(val_acc)
plt.grid()
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['acc','val_acc'], loc = 4)
```

Out[10]: <matplotlib.legend.Legend at 0x1a406a05940>



In [11]:
```
loss = hist_his['loss']
val_loss = hist_his['val_loss']
plt.plot(loss)
plt.plot(val_loss)
plt.grid()
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['loss','val_loss'], loc = 1)
```

Out[11]: <matplotlib.legend.Legend at 0x1a41f41f8d0>



In [12]:
```
print(loss)
print(val_loss)
print(acc)
print(val_acc)
```

```
[0.9289161065353615, 0.7224967200677879, 0.6759471297264099, 0.6306312879
418382, 0.5615544472959372, 0.5874900020161035, 0.556419783779005, 0.5151
083443018765, 0.5072679424308019, 0.5212524245944944, 0.5445805789423314,
0.4830270196947426, 0.45990993905960037, 0.45074576311883063, 0.440009480
75721463, 0.45428610602301533, 0.43651072863526275, 0.4407050690088089,
0.45363716278594024, 0.4122254128826063]
[0.7035112217432116, 0.5645308351007904, 0.7017756362513798, 0.5393307526
118871, 0.5456454886532411, 0.4796854878616769, 0.5120801929293609, 0.648
7705882911275, 0.469417158241679, 0.4875728570651717, 0.4746148335497553
7, 0.44605547989287025, 0.5734203847624907, 0.49476630040785163, 0.475654
5166416866, 0.43909876557385047, 0.4941506394889297, 0.5017719225185674,
0.5036047856982161, 0.4576926866500843]
[0.57054454, 0.68502474, 0.7289604, 0.7450495, 0.7815594, 0.77846533, 0.7
7970296, 0.7970297, 0.8032178, 0.7883663, 0.7945545, 0.8162129, 0.826113
9, 0.8304455, 0.8298267, 0.82054454, 0.8446782, 0.84158415, 0.83106434,
0.84158415]
[0.7263682, 0.7562189, 0.69154227, 0.78855723, 0.78109455, 0.8109453, 0.7
761194, 0.78855723, 0.8208955, 0.7960199, 0.8034826, 0.82835823, 0.778606
95, 0.80099505, 0.8208955, 0.83830845, 0.80845773, 0.79850745, 0.8059701
3, 0.8109453]
```

In [ ]:

Second validation of test2

Still, this part repeat the test2 model second time to verify the modle may work well on 30 classes.

The results of this repeat goes well. Thus, we finally confirm using this model for classifying the whole 30 classes by 120 epochs.

In [1]:
```
import numpy as np
import os
from tensorflow.keras import applications
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import optimizers, regularizers
from tensorflow.keras.models import Sequential
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dropout, Flatten, Dense, GaussianNoise,
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())
```

```
[name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality {
}
incarnation: 4375126007341548165
, name: "/device:XLA_GPU:0"
device_type: "XLA_GPU"
memory_limit: 17179869184
locality {
}
incarnation: 12474301033500099600
physical_device_desc: "device: XLA_GPU device"
, name: "/device:XLA_CPU:0"
device_type: "XLA_CPU"
memory_limit: 17179869184
locality {
}
incarnation: 13806675105940690515
physical_device_desc: "device: XLA_CPU device"
, name: "/device:GPU:0"
device_type: "GPU"
memory_limit: 15856484352
locality {
  bus_id: 1
  links {
  }
}
incarnation: 9834537617546067114
physical_device_desc: "device: 0, name: Tesla P100-PCIE-16GB, pci bus id:
0000:00:04.0, compute capability: 6.0"
]
```

In [3]:
```
import tensorflow.keras.backend as K
K.clear_session()
```

In [4]:
```
nrow = 200
ncol = 200
nclass = 3
base_model = applications.VGG16(weights='imagenet', input_shape=(nrow,ncol,3
for layer in base_model.layers:
    layer.trainable = False

x = base_model.output
x = Flatten()(x)
x = GaussianNoise(0.1)(x)
x = Dropout(0.5)(x)
x = Dense(2304, activation = 'relu')(x) # 18432/4
x = GaussianNoise(0.1)(x) # add noise to mitigate overfitting (regularizatic
x = Dropout(0.5)(x)
x = Dense(288, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)
pred = Dense(nclass, activation='softmax',
             kernel_initializer='random_uniform',
             bias_initializer='random_uniform',
             bias_regularizer=regularizers.l2(0.01),
             name='predictions')(x)
model = Model(inputs=base_model.input, outputs=pred)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.5/dist-packages/tensorflo
w/python/ops/resource_variable_ops.py:435: colocate_with (from tensorflo
w.python.framework.ops) is deprecated and will be removed in a future ver
sion.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From /usr/local/lib/python3.5/dist-packages/tensorflo
w/python/keras/layers/core.py:143: calling dropout (from tensorflow.pytho
n.ops.nn_ops) with keep_prob is deprecated and will be removed in a futur
e version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1
- keep_prob`.
```

In [5]:
```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         (None, 200, 200, 3)       0         
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 200, 200, 64)      1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 200, 200, 64)      36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 100, 100, 64)      0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 100, 100, 128)     73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 100, 100, 128)     147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 50, 50, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 50, 50, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 50, 50, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 50, 50, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 25, 25, 256)       0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 25, 25, 512)       1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 25, 25, 512)       2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 25, 25, 512)       2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 12, 12, 512)       0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 12, 12, 512)       2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 12, 12, 512)       2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 12, 12, 512)       2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 6, 6, 512)         0         
_________________________________________________________________
flatten (Flatten)            (None, 18432)             0         
_________________________________________________________________
gaussian_noise (GaussianNois (None, 18432)             0         
_________________________________________________________________
dropout (Dropout)            (None, 18432)             0         
_________________________________________________________________
dense (Dense)                (None, 2304)              42469632  
_________________________________________________________________
gaussian_noise_1 (GaussianNo (None, 2304)              0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 2304)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 288)               663840    
_________________________________________________________________
```

```
batch_normalization_v1 (Batc (None, 288)              1152      
_________________________________________________________________
dropout_2 (Dropout)          (None, 288)              0         
_________________________________________________________________
predictions (Dense)          (None, 3)                867       
=================================================================
Total params: 57,850,179
Trainable params: 43,134,915
Non-trainable params: 14,715,264
_________________________________________________________________
```

In [6]:
```
train_data_dir = './images_train'
batch_size = 32
train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)
train_generator = train_datagen.flow_from_directory(
                        train_data_dir,
                        target_size=(nrow,ncol),
                        batch_size=batch_size,
                        class_mode='categorical')
```

```
Found 1616 images belonging to 3 classes.
```

In [7]:
```
test_data_dir = './images_test'
batch_size = 32
test_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)
test_generator = train_datagen.flow_from_directory(
                        test_data_dir,
                        target_size=(nrow,ncol),
                        batch_size=batch_size,
                        class_mode='categorical')
```

```
Found 402 images belonging to 3 classes.
```

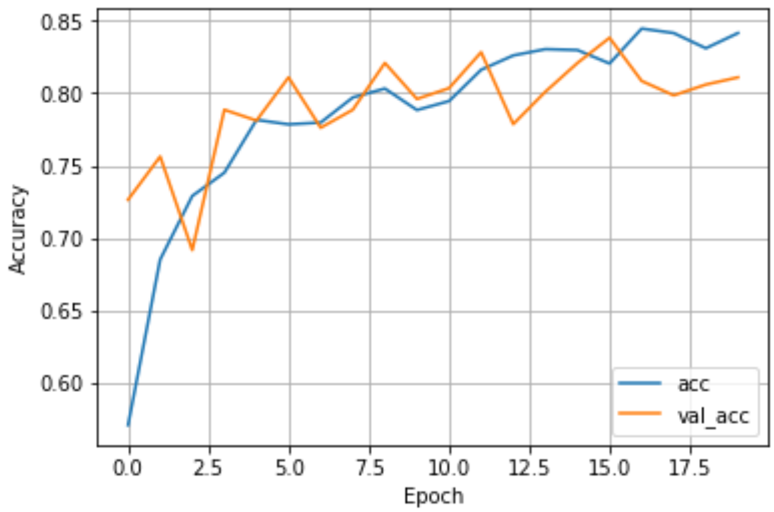
In [8]:
```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['a
steps_per_epoch = train_generator.n // batch_size
validation_steps = test_generator.n // batch_size
```

In [ ]:
```
nepochs = 120  # Number of epochs

# Call the fit_generator function
hist = model.fit_generator(
    train_generator,
    epochs=nepochs,
    steps_per_epoch=steps_per_epoch,
    validation_data=test_generator,
    validation_steps=validation_steps)
```

```
WARNING:tensorflow:From /usr/local/lib/python3.5/dist-packages/tensorflo
w/python/ops/math_ops.py:3066: to_int32 (from tensorflow.python.ops.math_
ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/120
13/13 [==============================] - 29s 2s/step - loss: 0.7350 - ac
c: 0.7488
51/51 [==============================] - 103s 2s/step - loss: 0.8890 - ac
c: 0.5928 - val_loss: 0.7350 - val_acc: 0.7488
Epoch 2/120
13/13 [==============================] - 29s 2s/step - loss: 0.6261 - ac
c: 0.7488
51/51 [==============================] - 89s 2s/step - loss: 0.6995 - ac
c: 0.7116 - val_loss: 0.6261 - val_acc: 0.7488
Epoch 3/120
15/51 [=======>......................] - ETA: 15s - loss: 0.5960 - acc:
0.7563
```

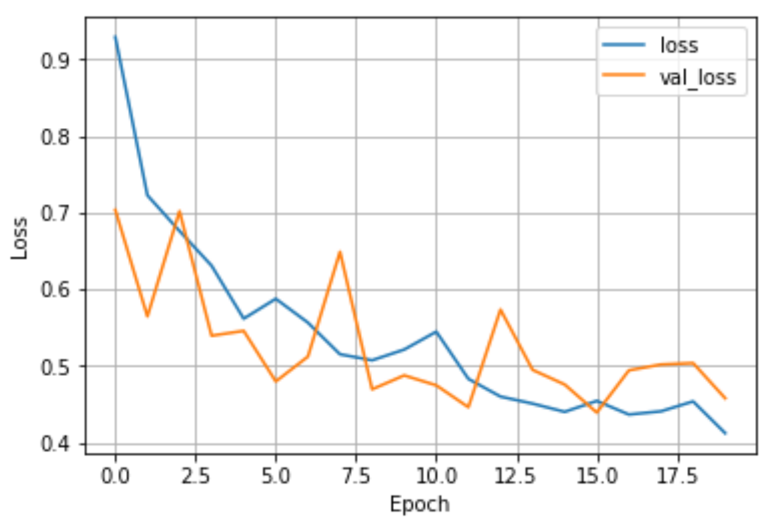
In [10]:
```
hist_his = hist.history
acc = hist_his['acc']
val_acc = hist_his['val_acc']
plt.plot(acc)
plt.plot(val_acc)
plt.grid()
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['acc','val_acc'], loc = 4)
```

Out[10]: <matplotlib.legend.Legend at 0x1a406a05940>



In [11]:
```
loss = hist_his['loss']
val_loss = hist_his['val_loss']
plt.plot(loss)
plt.plot(val_loss)
plt.grid()
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['loss','val_loss'], loc = 1)
```

Out[11]: <matplotlib.legend.Legend at 0x1a41f41f8d0>



In [12]:
```
print(loss)
print(val_loss)
print(acc)
print(val_acc)
```

```
[0.9289161065353615, 0.7224967200677879, 0.6759471297264099, 0.6306312879
418382, 0.5615544472959372, 0.5874900020161035, 0.556419783779005, 0.5151
083443018765, 0.5072679424308019, 0.5212524245944944, 0.5445805789423314,
0.4830270196947426, 0.45990993905960037, 0.45074576311883063, 0.440009480
75721463, 0.45428610602301533, 0.43651072863526275, 0.4407050690088089,
0.45363716278594024, 0.4122254128826063]
[0.7035112217432116, 0.5645308351007904, 0.7017756362513798, 0.5393307526
118871, 0.545645488653241l, 0.4796854878616769, 0.5120801929293609, 0.648
7705882911275, 0.469417158241679, 0.4875728570651717, 0.4746148335497553
7, 0.44605547989287025, 0.5734203847624907, 0.49476630040785163, 0.475654
5166416866, 0.43909876557385047, 0.4941506394889297, 0.5017719225185674,
0.5036047856982161, 0.4576926866500843]
[0.57054454, 0.68502474, 0.7289604, 0.7450495, 0.7815594, 0.77846533, 0.7
7970296, 0.7970297, 0.8032178, 0.7883663, 0.7945545, 0.8162129, 0.826113
9, 0.8304455, 0.8298267, 0.82054454, 0.8446782, 0.84158415, 0.83106434,
0.84158415]
[0.7263682, 0.7562189, 0.69154227, 0.78855723, 0.78109455, 0.8109453, 0.7
761194, 0.78855723, 0.8208955, 0.7960199, 0.8034826, 0.82835823, 0.778606
95, 0.80099505, 0.8208955, 0.83830845, 0.80845773, 0.79850745, 0.8059701
3, 0.8109453]
```

In [ ]:

Test (3 classes) results summary

For this part, we compare the results of all test models for 3 classes: untuned, test1, test2, and test3. We will use the `loss` , `val_loss` , `acc` , and `val_acc` from former tests.

For these models,

- `untuned` and `test1` are overfitting.
- `test2` is the best fitting.
- `test3` is underfitting.

Thus, we will use the model from test2 to classify the 30 classes.

In [1]:
```
untuned_loss = [0.7509350723559314, 0.4501491076875441, 0.33513500932419654,
untuned_val_loss = [0.5346578520077926, 0.49634360120846677, 0.5241671640139
untuned_acc = [0.6912129, 0.8273515, 0.8818069, 0.8824257, 0.9232673, 0.9164
untuned_val_acc = [0.76616913, 0.80845773, 0.8159204, 0.8308458, 0.81840795,
```

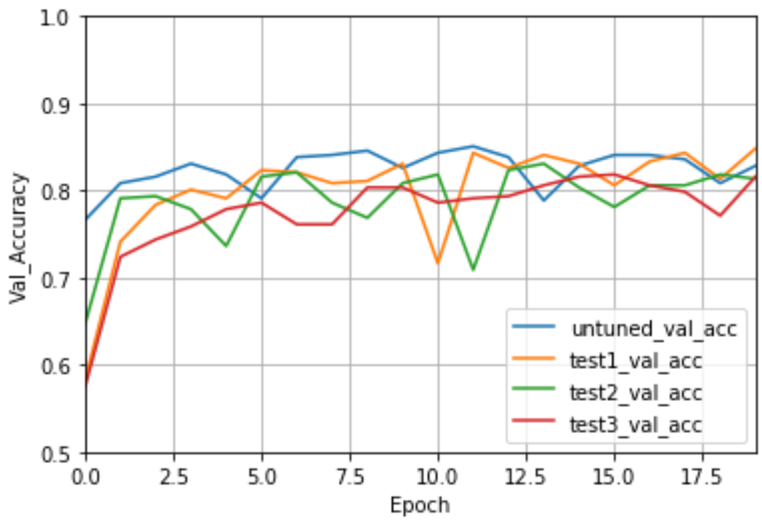
In [2]:
```
test1_loss = [0.9197668498105341, 0.6389658262233923, 0.5544988536598658, 0.
test1_val_loss = [1.2577795065366304, 0.6699514022240272, 0.5406953211014087
test1_acc = [0.58106434, 0.740099, 0.7642327, 0.79950494, 0.8279703, 0.83477
test1_val_acc = [0.58208954, 0.74129355, 0.7835821, 0.80099505, 0.7910448, 0
```

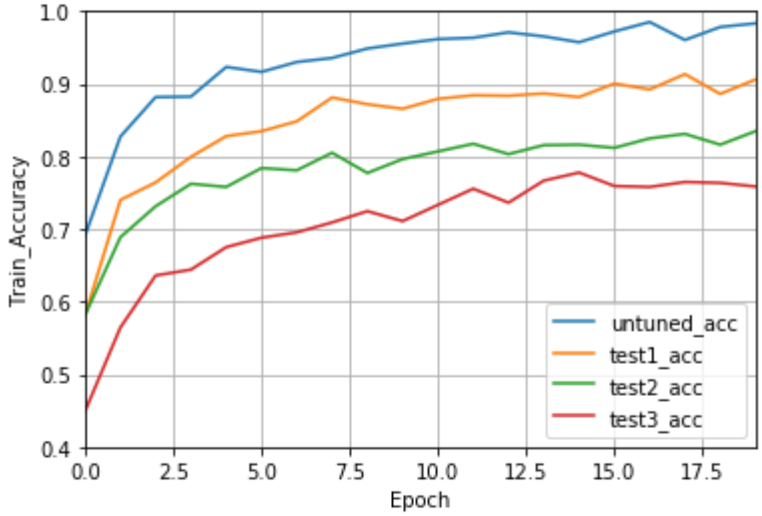
In [3]:
```
test2_loss = [0.9074013190030461, 0.7389471186970425, 0.6464045981295629, 0.
test2_val_loss = [0.8013866336607351, 0.5260722466358324, 0.4967596425515849
test2_acc = [0.58168316, 0.68873763, 0.73143566, 0.76237625, 0.75804454, 0.7
test2_val_acc = [0.6467662, 0.7910448, 0.7935323, 0.77860695, 0.7363184, 0.8
```

In [4]:
```
test3_loss = [1.0747226900651607, 0.9344037247286869, 0.8271783235953143, 0.
test3_val_loss = [0.8664734724127217, 0.6373515651549821, 0.6223491394409427
test3_acc = [0.44925743, 0.56435645, 0.6361386, 0.64418316, 0.67512375, 0.68
test3_val_acc = [0.57462686, 0.7238806, 0.7437811, 0.75870645, 0.77860695, 0
```

In [7]:

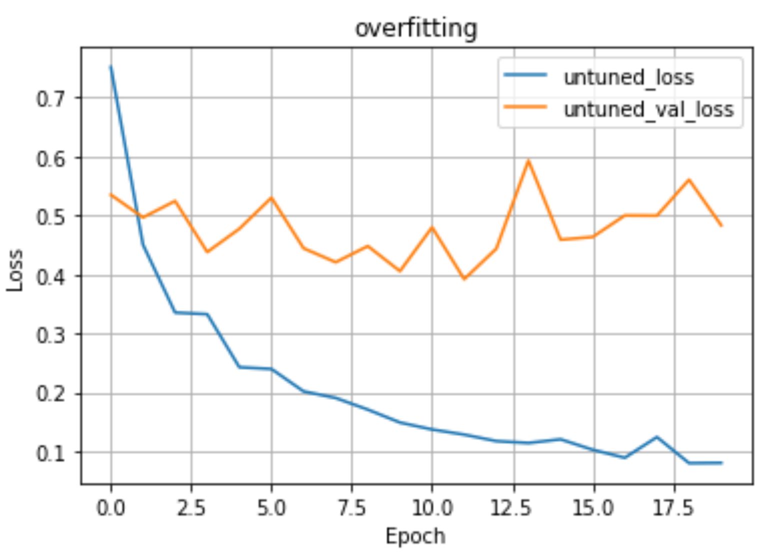
```
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline

plt.plot(untuned_val_acc)
plt.plot(test1_val_acc)
plt.plot(test2_val_acc)
plt.plot(test3_val_acc)
plt.grid()
plt.axis([0,19,0.5,1])
plt.xlabel('Epoch')
plt.ylabel('Val_Accuracy')
plt.legend(['untuned_val_acc','test1_val_acc','test2_val_acc','test3_val_acc
plt.show()

plt.plot(untuned_acc)
plt.plot(test1_acc)
plt.plot(test2_acc)
plt.plot(test3_acc)
plt.grid()
plt.axis([0,19,0.4,1])
plt.xlabel('Epoch')
plt.ylabel('Train_Accuracy')
plt.legend(['untuned_acc','test1_acc','test2_acc','test3_acc'], loc = 4)
plt.show()
```

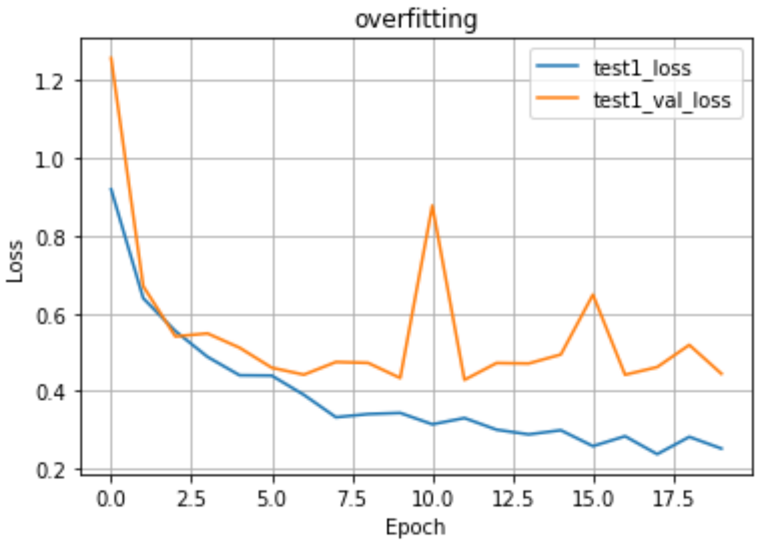


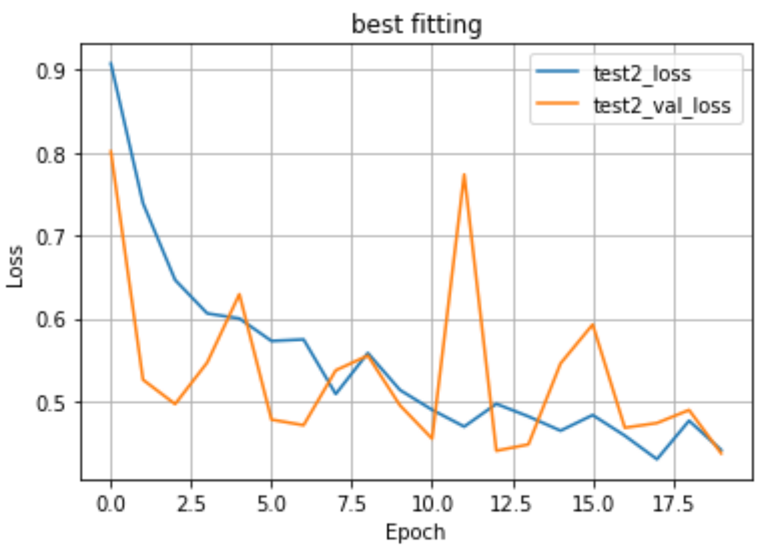


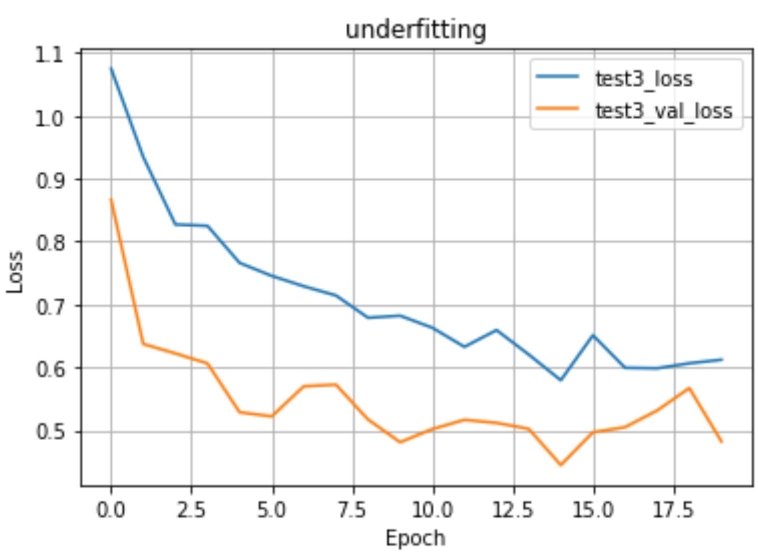
Train_Accuracy
Epoch
untuned_acc
test1_acc
test2_acc
test3_acc
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

In [13]:
```
plt.plot(untuned_loss)
plt.plot(untuned_val_loss)
plt.title('overfitting')
plt.grid()
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['untuned_loss','untuned_val_loss'], loc = 1)
plt.show()

plt.plot(test1_loss)
plt.plot(test1_val_loss)
plt.title('overfitting')
plt.grid()
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['test1_loss','test1_val_loss'], loc = 1)
plt.show()

plt.plot(test2_loss)
plt.plot(test2_val_loss)
plt.title('best fitting')
plt.grid()
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['test2_loss','test2_val_loss'], loc = 1)
plt.show()

plt.plot(test3_loss)
plt.plot(test3_val_loss)
plt.title('underfitting')
plt.grid()
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['test3_loss','test3_val_loss'], loc = 1)
plt.show()
```




overfitting
test1_loss
test1_val_loss
Loss
Epoch
1.2
1.0
0.8
0.6
0.4
0.2
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


best fitting
test2_loss
test2_val_loss
Loss
Epoch
0.9
0.8
0.7
0.6
0.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


underfitting
test3_loss
test3_val_loss
Loss
Epoch
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

In [ ]:

Run VGG16 on 30 classes with 120 epochs

Finally, we use the model from test2 to classify 30 artists' paintings with 120 epochs.

Obviously, the validation accuraccy is stable at more than %60, and the validation loss keep decreasing during the 120 epochs while keep reasonable difference between the training loss.

Given that there are 30 classes and some of the artists have similar art style, the accuracy of 60% is an acceptable result.

In [1]:
```
import numpy as np
import os
from tensorflow.keras import applications
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import optimizers, regularizers
from tensorflow.keras.models import Sequential
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dropout, Flatten, Dense, GaussianNoise,
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())
```

```
[name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality {
}
incarnation: 17738829143079950615
, name: "/device:GPU:0"
device_type: "GPU"
memory_limit: 2264907776
locality {
  bus_id: 1
  links {
  }
}
incarnation: 18079149120323357725
physical_device_desc: "device: 0, name: GeForce GTX 970M, pci bus id: 000
0:01:00.0, compute capability: 5.2"
]
```

In [3]:
```
import tensorflow.keras.backend as K
K.clear_session()
```

In [4]:
```
nrow = 200
ncol = 200
nclass = 30
base_model = applications.VGG16(weights='imagenet', input_shape=(nrow,ncol,3
for layer in base_model.layers:
    layer.trainable = False

x = base_model.output
x = Flatten()(x)
x = GaussianNoise(0.1)(x)
x = Dropout(0.5)(x)
x = Dense(2304, activation = 'relu')(x) # 18432/4
x = GaussianNoise(0.1)(x) # add noise to mitigate overfitting (regularizatio
x = Dropout(0.5)(x)
x = Dense(288, activation='relu')(x)
x = BatchNormalization()(x)
x = Dropout(0.5)(x)
pred = Dense(nclass, activation='softmax',
             kernel_initializer='random_uniform',
             bias_initializer='random_uniform',
             bias_regularizer=regularizers.l2(0.01),
             name='predictions')(x)
model = Model(inputs=base_model.input, outputs=pred)
```

```
WARNING:tensorflow:From D:\Anaconda\envs\tensorflow\lib\site-packages\ten
sorflow\python\ops\resource_variable_ops.py:435: colocate_with (from tens
orflow.python.framework.ops) is deprecated and will be removed in a futur
e version.
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From D:\Anaconda\envs\tensorflow\lib\site-packages\ten
sorflow\python\keras\layers\core.py:143: calling dropout (from tensorflo
w.python.ops.nn_ops) with keep_prob is deprecated and will be removed in
a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1
- keep_prob`.
```

In [5]:
```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
input_1 (InputLayer)         (None, 200, 200, 3)       0         
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 200, 200, 64)      1792      
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 200, 200, 64)      36928     
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 100, 100, 64)      0         
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 100, 100, 128)     73856     
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 100, 100, 128)     147584    
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 50, 50, 128)       0         
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 50, 50, 256)       295168    
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 50, 50, 256)       590080    
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 50, 50, 256)       590080    
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 25, 25, 256)       0         
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 25, 25, 512)       1180160   
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 25, 25, 512)       2359808   
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 25, 25, 512)       2359808   
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 12, 12, 512)       0         
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 12, 12, 512)       2359808   
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 12, 12, 512)       2359808   
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 12, 12, 512)       2359808   
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 6, 6, 512)         0         
_________________________________________________________________
flatten (Flatten)            (None, 18432)             0         
_________________________________________________________________
gaussian_noise (GaussianNois (None, 18432)             0         
_________________________________________________________________
dropout (Dropout)            (None, 18432)             0         
_________________________________________________________________
dense (Dense)                (None, 2304)              42469632  
_________________________________________________________________
gaussian_noise_1 (GaussianNo (None, 2304)              0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 2304)              0         
_________________________________________________________________
dense_1 (Dense)              (None, 288)               663840    
_________________________________________________________________
```

```
batch_normalization_v1 (Batc (None, 288)               1152
_________________________________________________________________
dropout_2 (Dropout)          (None, 288)               0
_________________________________________________________________
predictions (Dense)          (None, 30)                8670
=================================================================
Total params: 57,857,982
Trainable params: 43,142,718
Non-trainable params: 14,715,264
_________________________________________________________________
```

In [6]:

```
train_data_dir = './images_train'
batch_size = 5
train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                  zoom_range=0.2,
                                   horizontal_flip=True)
train_generator = train_datagen.flow_from_directory(
                        train_data_dir,
                        target_size=(nrow,ncol),
                        batch_size=batch_size,
                        class_mode='categorical')
```

```
Found 5687 images belonging to 30 classes.
```

In [7]:

```
test_data_dir = './images_test'
batch_size = 5
test_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                  zoom_range=0.2,
                                   horizontal_flip=True)
test_generator = train_datagen.flow_from_directory(
                        test_data_dir,
                        target_size=(nrow,ncol),
                        batch_size=batch_size,
                        class_mode='categorical')
```

```
Found 1404 images belonging to 30 classes.
```

In [8]:

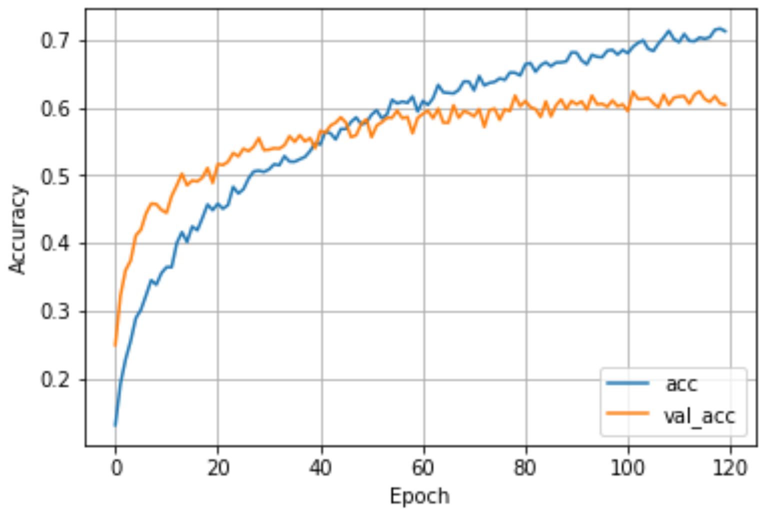
```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['a
steps_per_epoch = train_generator.n // batch_size
validation_steps = test_generator.n // batch_size
```

In [9]:
```
nepochs = 120  # Number of epochs

# Call the fit_generator function
hist = model.fit_generator(
    train_generator,
    epochs=nepochs,
    steps_per_epoch=steps_per_epoch,
    validation_data=test_generator,
    validation_steps=validation_steps)
```

```
WARNING:tensorflow:From D:\Anaconda\envs\tensorflow\lib\site-packages\ten
sorflow\python\ops\math_ops.py:3066: to_int32 (from tensorflow.python.op
s.math_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.cast instead.
Epoch 1/120
281/281 [==============================] - 39s 137ms/step - loss: 2.5986
- acc: 0.2493
1138/1138 [==============================] - 264s 232ms/step - loss: 3.14
91 - acc: 0.1310 - val_loss: 2.5986 - val_acc: 0.2493
Epoch 2/120
281/281 [==============================] - 32s 114ms/step - loss: 2.4152
- acc: 0.3219
1138/1138 [==============================] - 186s 164ms/step - loss: 2.85
32 - acc: 0.1918 - val_loss: 2.4152 - val_acc: 0.3219
Epoch 3/120
281/281 [==============================] - 31s 110ms/step - loss: 2.2909
- acc: 0.3597
1138/1138 [==============================] - 187s 165ms/step - loss: 2.69
```

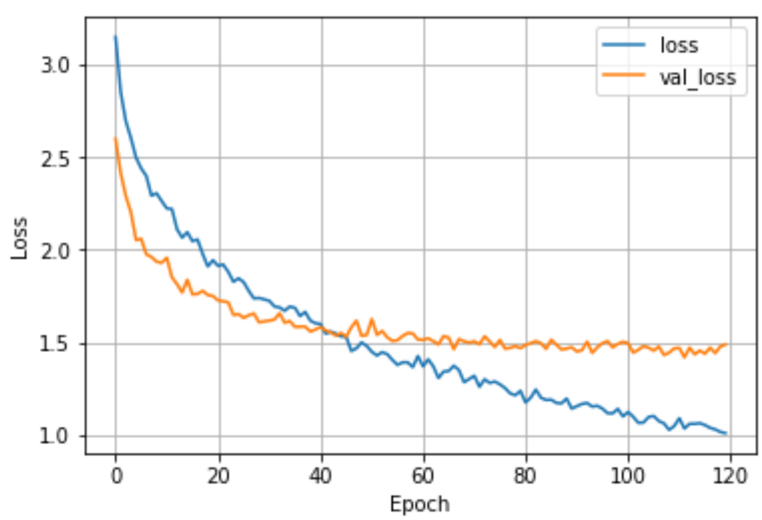
In [10]:
```
hist_his = hist.history
acc = hist_his['acc']
val_acc = hist_his['val_acc']
plt.plot(acc)
plt.plot(val_acc)
plt.grid()
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['acc','val_acc'], loc = 4)
```

Out[10]: <matplotlib.legend.Legend at 0x24bb18c3748>



In [11]:
```
loss = hist_his['loss']
val_loss = hist_his['val_loss']
plt.plot(loss)
plt.plot(val_loss)
plt.grid()
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['loss','val_loss'], loc = 1)
```

Out[11]: <matplotlib.legend.Legend at 0x24bb1909ba8>



In [12]:
```
print(loss)
print(val_loss)
print(acc)
print(val_acc)
```

```
[3.14887666830137, 2.8532378607524054, 2.6948824729885055, 2.603927112807
9124, 2.4979331421164566, 2.440019819100083, 2.399272446737613, 2.2927422
82639536, 2.30581997699429, 2.2634986593640707, 2.2238892891274116, 2.220
4405017470945, 2.110854354446794, 2.0661793674560305, 2.0953340293737712,
2.047150615105546, 2.0561495057272956, 1.9780113589623198, 1.910356094107
9305, 1.9441882107470394, 1.9133827449442402, 1.921118175116662, 1.881087
851315784, 1.8279229860241224, 1.8463588798163497, 1.825864585407677, 1.7
781568921490631, 1.7375602504637742, 1.7402194728693268, 1.73256280559719
56, 1.7249106890065862, 1.69454954486751, 1.6893229365626559, 1.671006831
0514494, 1.694082800954638, 1.687026359556177, 1.642625775837366, 1.66534
71243717937, 1.6179302782737373, 1.6038959688491135, 1.5998659210237989,
1.547910915263533, 1.5579253501190502, 1.5476253076022677, 1.532900247846
6693, 1.5308413288705227, 1.4540808080568326, 1.4685977000107093, 1.50013
6630329596, 1.481121332802734, 1.4495783738067454, 1.429472905664415, 1.4
4756361214483, 1.4370123727857302, 1.406100361045769, 1.3790945432122508,
1.392482722140436, 1.3928008942883012, 1.3669300967664004, 1.426529755423
8139, 1.3708891403490093, 1.409018208908976, 1.3711955135172986, 1.308138
2734742633, 1.3428455541588016, 1.346025437286937, 1.3744070033800084, 1.
3512422148448124, 1.2855672028039522, 1.302735023932862, 1.32019892789021
26, 1.2604598219240981, 1.3012864021746517, 1.2816157268115596, 1.2891119
165122982, 1.2739969535231463, 1.253721626086679, 1.2249442387456757, 1.2
151649425905744, 1.2389106319423655, 1.1761944099647603, 1.20193574727315
24, 1.2460170154629597, 1.2017211732416113, 1.1896493020401135, 1.1911034
554587645, 1.173888971134614, 1.1710680727395752, 1.1973944686435742, 1.1
431129749884648, 1.1558843259851124, 1.168114076966411, 1.173178361045238
9, 1.1538946076419256, 1.1581114742473222, 1.1429505718009574, 1.11822127
09685542, 1.1175019128334236, 1.1415150390097164, 1.1008182604233347, 1.1
248400598477506, 1.0991875380395544, 1.0664902638233917, 1.06784067309230
8, 1.097637352785343, 1.1032167018122783, 1.0760066149307927, 1.063404732
6620217, 1.0284685007055279, 1.0486837283565518, 1.0912402716370475, 1.03
56118842272908, 1.0608548783499525, 1.0607053105070392, 1.064400925454134
2, 1.0540254852266635, 1.0391940124246437, 1.0307174821649383, 1.01709449
79791328, 1.0105065652225786]
[2.5986390673817263, 2.4152131207897147, 2.2908821453827555, 2.2023981801
13144, 2.0528401929712805, 2.0596567402955053, 1.9760068330476293, 1.9615
514961425944, 1.936446952438015, 1.9301648468733683, 1.9580832193754747,
1.851916514999926, 1.8150155751094275, 1.7698578359393462, 1.838367413371
4831, 1.7600538260253722, 1.7621840319175313, 1.7785802632570267, 1.75639
4695556885, 1.7512089908653308, 1.7272833022570695, 1.7224288245537103,
1.716591116796609, 1.6486663927599203, 1.6531337413181189, 1.632963371658
6647, 1.6471444270788986, 1.6565643461997823, 1.6086759656879825, 1.61392
50046895068, 1.6181251450683722, 1.623818162276651, 1.6580979210122413,
1.6068940174325081, 1.6154659823166517, 1.584228739460592, 1.585238944460
701, 1.5855607588202079, 1.5594624436834954, 1.5699501504253237, 1.582968
9502716064, 1.5645903365361733, 1.55543631887945, 1.536528892205279, 1.55
11978027341204, 1.5310094711301165, 1.581782923182771, 1.617809724500170
8, 1.5361509071497306, 1.5404548884180518, 1.6265546720859418, 1.54115970
09734963, 1.5621406999930367, 1.5277147977563197, 1.5086268228064441, 1.5
118827614601822, 1.5350136824243859, 1.5507322604373575, 1.54830767573570
02, 1.5157892402539899, 1.5139411654044936, 1.522371015594311, 1.50671618
14346115, 1.4893123643121473, 1.5346368134843709, 1.5262638342125985, 1.4
637971964276983, 1.5183051515940669, 1.5058256992241665, 1.49980634092966
```

```
43, 1.5059065040369979, 1.4899178577911811, 1.5330901763922273, 1.5056956
097536665, 1.4753895334342622, 1.5136543880233349, 1.4666676466681354, 1.
470921202804694 3, 1.480322968705269, 1.4692638841228978, 1.48668276256089
54, 1.495926550884476, 1.5064830970933853, 1.4961913378107166, 1.46541278
42749989, 1.5147791323500597, 1.488247580905826, 1.4624823325999692, 1.46
66374412454743, 1.474243866982356, 1.4511621594429016, 1.458440694706083
5, 1.5042794497279297, 1.4436638161964264, 1.474070118972203, 1.496235860
1205714, 1.5070647389969367, 1.4736667002064052, 1.4920057622828518, 1.50
3540256428549, 1.4964379253548659, 1.4452120655637195, 1.460496266276386
7, 1.4804624406309315, 1.4719022546639646, 1.4560790446676393, 1.47778608
70100956, 1.432703056488703, 1.444938014781772, 1.4677913017650517, 1.470
7328780584064, 1.4194630855343096, 1.4723218056601985, 1.436070863939689,
1.4563526885047078, 1.439350330294027, 1.4723250337634435, 1.442419101520
5774, 1.4782642004116575, 1.4890809308843576]
[0.13100053, 0.19184104, 0.228064, 0.25531915, 0.28890452, 0.30138913, 0.
32266572, 0.34499738, 0.33831546, 0.3553719, 0.3643397, 0.36416388, 0.399
85934, 0.41586074, 0.40144187, 0.42430103, 0.41867417, 0.4369615, 0.45683
137, 0.44839108, 0.4578864, 0.45050114, 0.45595217, 0.48285565, 0.473008
6, 0.47933885, 0.49516442, 0.50553894, 0.50676984, 0.5048356, 0.50870407,
0.51626515, 0.5146826, 0.52822226, 0.5201336, 0.5199578, 0.5236504, 0.526
99137, 0.5356075, 0.5473888, 0.54545456, 0.5646211, 0.56180763, 0.552839
8, 0.5681379, 0.5683137, 0.57868826, 0.5853701, 0.57481974, 0.5776332, 0.
5894145, 0.5959205, 0.58396345, 0.58976614, 0.6113944, 0.6059434, 0.60875
684, 0.60682255, 0.6163179, 0.5939863, 0.60981184, 0.6038333, 0.6136803,
0.63337433, 0.62229645, 0.6215931, 0.6207139, 0.6259891, 0.6384737, 0.638
29786, 0.62528574, 0.6463865, 0.63249516, 0.6363636, 0.63777035, 0.643045
54, 0.64075965, 0.6516617, 0.65131, 0.6472657, 0.66379464, 0.66572887, 0.
6527167, 0.6623879, 0.66678387, 0.6609812, 0.6664322, 0.66660804, 0.66801
476, 0.6813786, 0.6810269, 0.6694215, 0.66379464, 0.6778618, 0.6748725,
0.6741692, 0.68401617, 0.68507123, 0.6782135, 0.6848954, 0.6801477, 0.689
643, 0.6954458, 0.6991384, 0.68647796, 0.68348867, 0.6947424, 0.70300686,
0.7133814, 0.7008968, 0.6965008, 0.7086337, 0.698435, 0.6977317, 0.703358
53, 0.7014243, 0.70388603, 0.71478814, 0.71654654, 0.712678]
[0.24928775, 0.32193732, 0.3596866, 0.37393162, 0.41096866, 0.41951567,
0.4437322, 0.4579772, 0.45726496, 0.44871795, 0.44444445, 0.46937323, 0.4
8504272, 0.50213677, 0.48504272, 0.49216524, 0.49074075, 0.49643874, 0.51
06838, 0.48860398, 0.51638174, 0.51495725, 0.519943, 0.53276354, 0.527777
8, 0.5391738, 0.5356125, 0.54202276, 0.5548433, 0.537037, 0.5377493, 0.53
988606, 0.5391738, 0.54415953, 0.5576923, 0.5491453, 0.5591168, 0.5498575
6, 0.5548433, 0.53988606, 0.5655271, 0.56267804, 0.5733618, 0.57763535,
0.5854701, 0.57763535, 0.5562678, 0.5591168, 0.5762108, 0.5826211, 0.5562
678, 0.5733618, 0.57834756, 0.58475786, 0.58475786, 0.5954416, 0.5840456,
0.58618236, 0.5619658, 0.58475786, 0.59045583, 0.5954416, 0.5840456, 0.59
82906, 0.57763535, 0.57763535, 0.6032764, 0.58475786, 0.59472936, 0.59188
03, 0.5868946, 0.59757835, 0.57122505, 0.59615386, 0.5982906, 0.5819088,
0.59757835, 0.59472936, 0.61752135, 0.6018519, 0.6096866, 0.6004273, 0.59
757835, 0.58618236, 0.60897434, 0.5868946, 0.6032764, 0.6118234, 0.597578
35, 0.6096866, 0.60470086, 0.60897434, 0.5968661, 0.61752135, 0.60470086,
0.6054131, 0.6011396, 0.6103989, 0.6018519, 0.60470086, 0.59472936, 0.623
2194, 0.6125356, 0.6125356, 0.6139601, 0.60612535, 0.6004273, 0.6189459,
0.6039886, 0.61467236, 0.61609685, 0.6168091, 0.6054131, 0.6189459, 0.623
93165, 0.6125356, 0.6082621, 0.6168091, 0.60612535, 0.6039886]
```

In [ ]:

30 classes results summary

For this part, we compare the results two models for 3 classes: untuned, and the model from test2. We will use the `loss` , `val_loss` , `acc` , and `val_acc` from the former tests.

In conclusion,

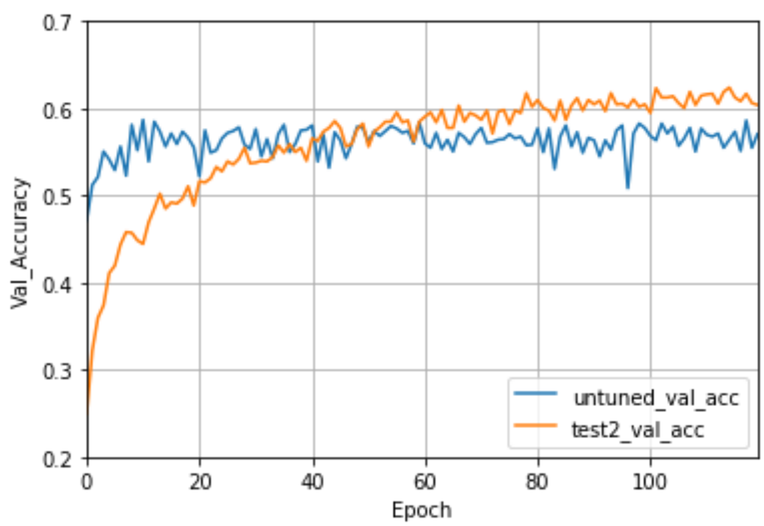
- `untuned` is overfitting.
- `test2 for 30 classes` mitigate the severe overfitting from untuned model. However, since the resource limitation, we only run 120 epochs. From the curves, `val_acc` does not seem to fully converge. Thus, the accuracy may get higher if adding more epochs.

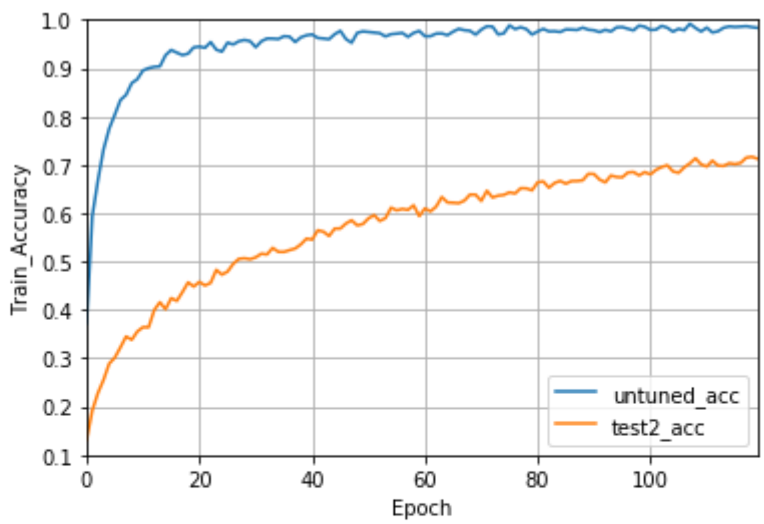
From the `untuned_loss_curve` , the `val_loss` keep increasing during all 120 epoch, which reflects severe overfitting; By comparison, from the `test2_loss_curve` , the `val_loss` keep decreasing during the 120 epochs which is what we expect.

In [1]:
```
untuned_loss = [2.3461001035050724, 1.4235979016874845, 1.1430863561457525,
untuned_val_loss = [1.8798491602594203, 1.8879662372849204, 1.88664359396154
untuned_acc = [0.3726042, 0.5936346, 0.66731143, 0.73184454, 0.77474946, 0.8
untuned_val_acc = [0.47150996, 0.51210827, 0.52136755, 0.5505698, 0.54131055
```

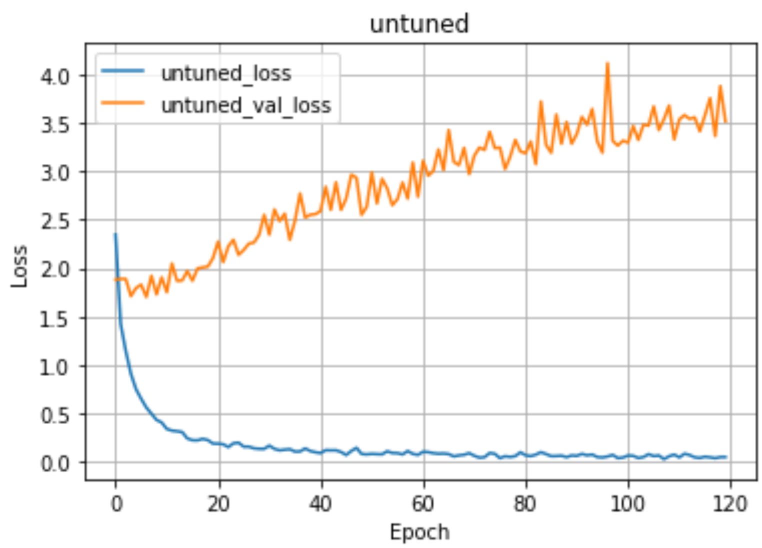
In [2]:
```
test2_loss = [3.14887666830137, 2.8532378607524054, 2.6948824729885055, 2.60
test2_val_loss = [2.5986390673817263, 2.4152131207897147, 2.29088214538275 55
test2_acc = [0.13100053, 0.19184104, 0.228064, 0.25531915, 0.28890452, 0.301
test2_val_acc = [0.24928775, 0.32193732, 0.3596866, 0.37393162, 0.41096866,
```

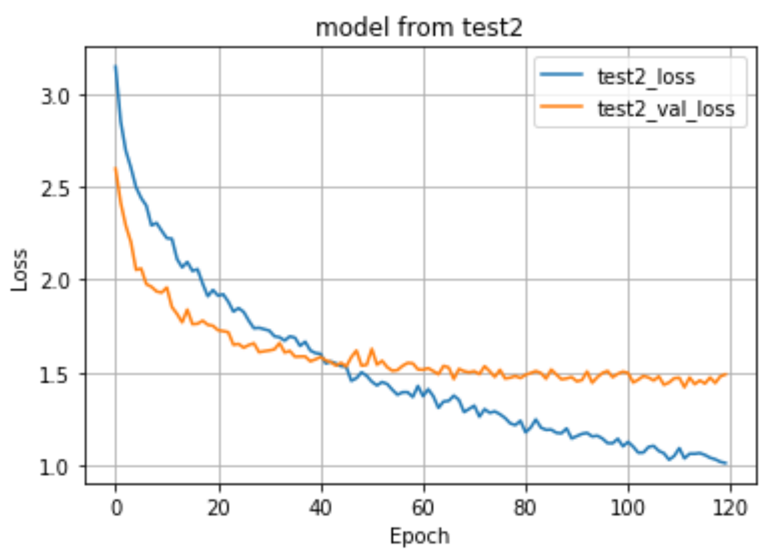
In [6]:
```
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline

plt.plot(untuned_val_acc)
plt.plot(test2_val_acc)
plt.grid()
plt.axis([0,119,0.2,0.7])
plt.xlabel('Epoch')
plt.ylabel('Val_Accuracy')
plt.legend(['untuned_val_acc','test2_val_acc'], loc = 4)
plt.show()

plt.plot(untuned_acc)
plt.plot(test2_acc)
plt.grid()
plt.axis([0,119,0.1,1])
plt.xlabel('Epoch')
plt.ylabel('Train_Accuracy')
plt.legend(['untuned_acc','test2_acc'], loc = 4)
plt.show()
```





In [8]:
```
plt.plot(untuned_loss)
plt.plot(untuned_val_loss)
plt.title('untuned')
plt.grid()
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['untuned_loss','untuned_val_loss'], loc = 2)
plt.show()

plt.plot(test2_loss)
plt.plot(test2_val_loss)
plt.title('model from test2')
plt.grid()
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['test2_loss','test2_val_loss'], loc = 1)
plt.show()
```





In [ ]: