Όραση Υπολογιστών

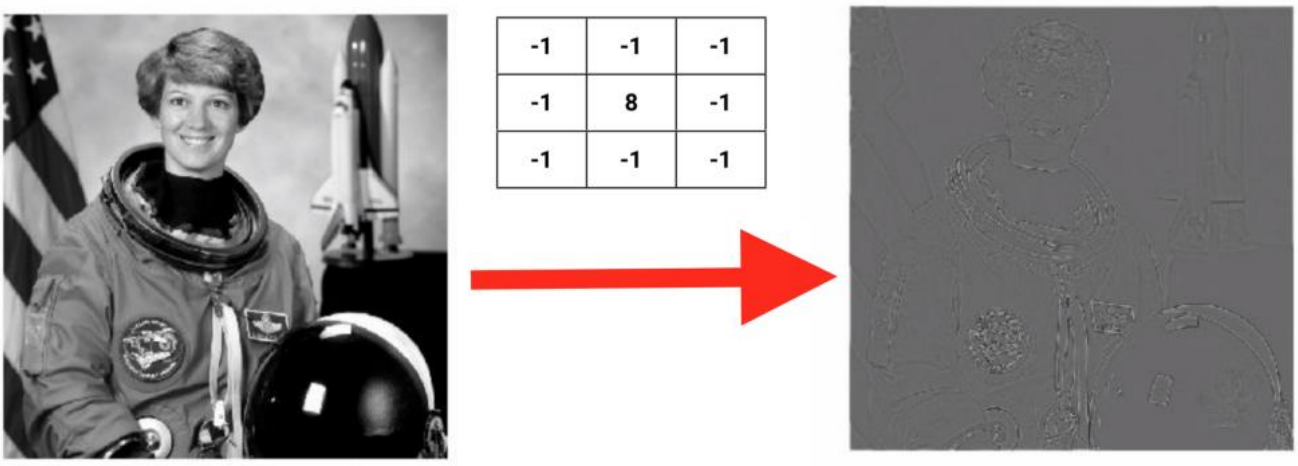
Εργασία 4η

Αλβανάκη Παρασκευή
ΑΜ 57286

Θεωρητικό υπόβαθρο

➢ CONVOLUTION

Μια συνέλιξη είναι ο τρόπος με τον οποίο η είσοδος τροποποιείται από ένα φίλτρο. Στα συνελικτικά δίκτυα, μεταφέρονται πολλαπλά φίλτρα για να κάνουν slide μέσω της εικόνας και να τα χαρτογραφούν ένα προς ένα τα σημεία και να μαθαίνουν διαφορετικά τμήματα μιας εικόνας εισόδου. Φανταστείτε ένα μικρό φίλτρο που ολισθαίνει από τα αριστερά προς τα δεξιά σε όλη την εικόνα από πάνω προς τα κάτω και το κινούμενο φίλτρο ψάχνει, ας πούμε, dark edge. Κάθε φορά που εντοπίζεται ένα ταίριασμα, έχει χαρτογραφιθεί σε μια εικόνα εξόδου.



2	0	1	1
0	1	0	0
0	0	1	0
0	3	0	0

An input image (no padding)

1	0	1
0	0	0
0	1	0

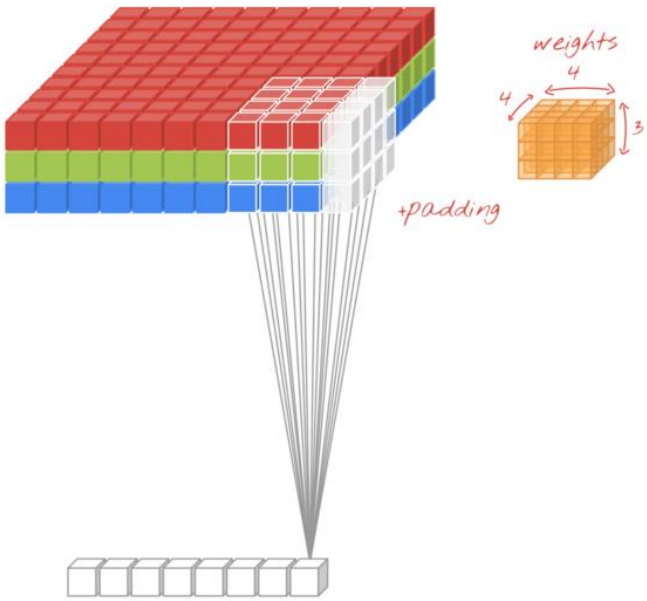
A filter (3x3)

3	2

Output image (after convolving with stride 1)

➢ PADDING

Αν θέλουμε να διατηρήσετε την εικόνα εξόδου στο ίδιο πλάτος και ύψος χωρίς να μειώσετε το μέγεθος του φίλτρου, μπορούμε να προσθέσουμε padding στην αρχική εικόνα με μηδέν και να φτιάξουμε ένα convolution slice μέσα στην εικόνα.



We can applying padding (zeros along the borders) to the original image to maintain the same output dimensions.

➢ FEATURE MAP

Ένα κανάλι εξόδου των συρραπτικών ονομάζεται χάρτης χαρακτηριστικών. Κωδικοποιεί την παρουσία ή την απουσία και τον βαθμό παρουσίας του χαρακτηριστικού που ανιχνεύει.

➢ POOLING

Η συγκέντρωση χρησιμοποιείται για τη μείωση του μεγέθους της εικόνας πλάτους και ύψους. Σημειώστε ότι το βάθος καθορίζεται από τον αριθμό των καναλιών. Όπως υποδηλώνει το όνομα, το μόνο που κάνει είναι ότι επιλέγει τη μέγιστη τιμή σε ένα συγκεκριμένο μέγεθος του παραθύρου. Παρόλο που εφαρμόζεται συνήθως χωρικά για να μειώσει τις διαστάσεις x, y μιας εικόνας.

➢ MAX-POOLING

Η μέγιστη συγκέντρωση χρησιμοποιείται για τη μείωση του μεγέθους της εικόνας, με τη χαρτογράφηση του μεγέθους ενός δεδομένου παραθύρου σε ένα μόνο αποτέλεσμα, λαμβάνοντας τη μέγιστη τιμή των στοιχείων στο παράθυρο.

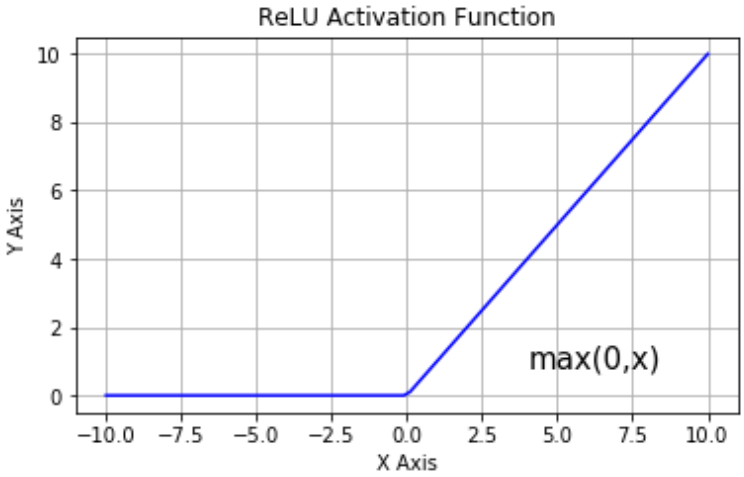
2	0	1	1
0	1	0	0
0	0	1	0
0	3	0	0

Max pooling with a 2x2 window and stride 2

2	1
3	1

➢ RELU

Το ReLU είναι η πλέον χρησιμοποιούμενη λειτουργία ενεργοποίησης στον κόσμο. Σήμερα, χρησιμοποιείται σε όλα σχεδόν τα συνελικτικά νευρωνικά δίκτυα ή στη βαθιά εκμάθηση. Μετά από κάθε στρώμα μετατροπής, είναι σύνηθες να εφαρμόζεται αμέσως μετά ένα μη γραμμικό στρώμα (ή στρώμα ενεργοποίησης).Ο σκοπός αυτού του στρώματος είναι να εισαγάγει μη γραμμικότητα σε ένα σύστημα το οποίο βασικά έχει μόλις υπολογίσει τις γραμμικές λειτουργίες κατά τη διάρκεια των στρώσεων conv (απλά πολλαπλασιασμός στοιχείων και άθροισμα).Στο παρελθόν, χρησιμοποιήθηκαν μη γραμμικές λειτουργίες όπως το tanh και το sigmoid, αλλά οι ερευνητές ανακάλυψαν ότι τα στρώματα ReLU λειτουργούν πολύ καλύτερα επειδή το δίκτυο είναι ικανό να εκπαιδεύεται πολύ πιο γρήγορα (λόγω της υπολογιστικής αποτελεσματικότητας) χωρίς να κάνει σημαντική διαφορά στην ακρίβεια. Βοηθά επίσης να εξαλύψει το προβλημα του vanishing gradient, το οποίο προκαλείται από το γεγονός ότι τα χαμηλότερα στρώματα του δικτύου προπονούνται πολύ αργά, επειδή η κλίση μειώνεται εκθετικά μέσα από τα στρώματα .Το στρώμα ReLU εφαρμόζει τη συνάρτηση f (x) = max (0, x) σε όλες τις τιμές στον όγκο εισόδου. Βασικά, αυτό το στρώμα αλλάζει απλώς όλες τις αρνητικές ενεργοποιήσεις σε 0. Αυτό το στρώμα αυξάνει τις μη γραμμικές ιδιότητες του μοντέλου και του συνολικού δικτύου χωρίς να επηρεάζει τα δεκτικά πεδία του στρώματος convolution.



➢ AUGMENTATION

Η αύξηση των δεδομένων εικόνας είναι μια τεχνική που μπορεί να χρησιμοποιηθεί για την τεχνητή επέκταση του μεγέθους ενός συνόλου δεδομένων εκπαίδευσης με τη δημιουργία τροποποιημένων εκδόσεων εικόνων στο σύνολο δεδομένων.

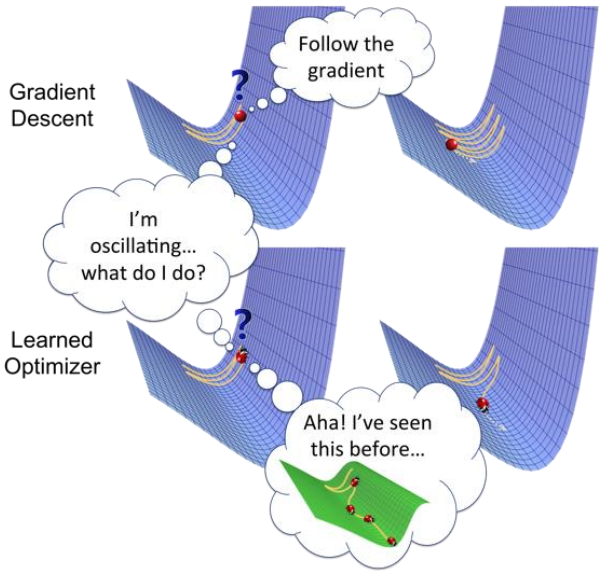
Η κατάρτιση των μοντέλων νευρωνικών δικτύων σε βάθος εκμάθησης σε περισσότερα δεδομένα μπορεί να οδηγήσει σε πιο εξειδικευμένα μοντέλα και οι τεχνικές επαύξησης μπορούν να δημιουργήσουν παραλλαγές των εικόνων που μπορούν να βελτιώσουν την ικανότητα των μοντέλων να γενικεύουν τι έχουν μάθει σε νέες εικόνες.

➢ CATEGORICAL CROSSENTROPY

Η categorical crossentropy είναι μια συνάρτηση απώλειας που χρησιμοποιείται για την κατηγοριοποίηση μιας ετικέτας. Αυτό συμβαίνει όταν ισχύει μόνο μία κατηγορία για κάθε σημείο δεδομένων. Με άλλα λόγια, ένα παράδειγμα μπορεί να ανήκει μόνο σε μία τάξη

➢ OPTIMIZATION

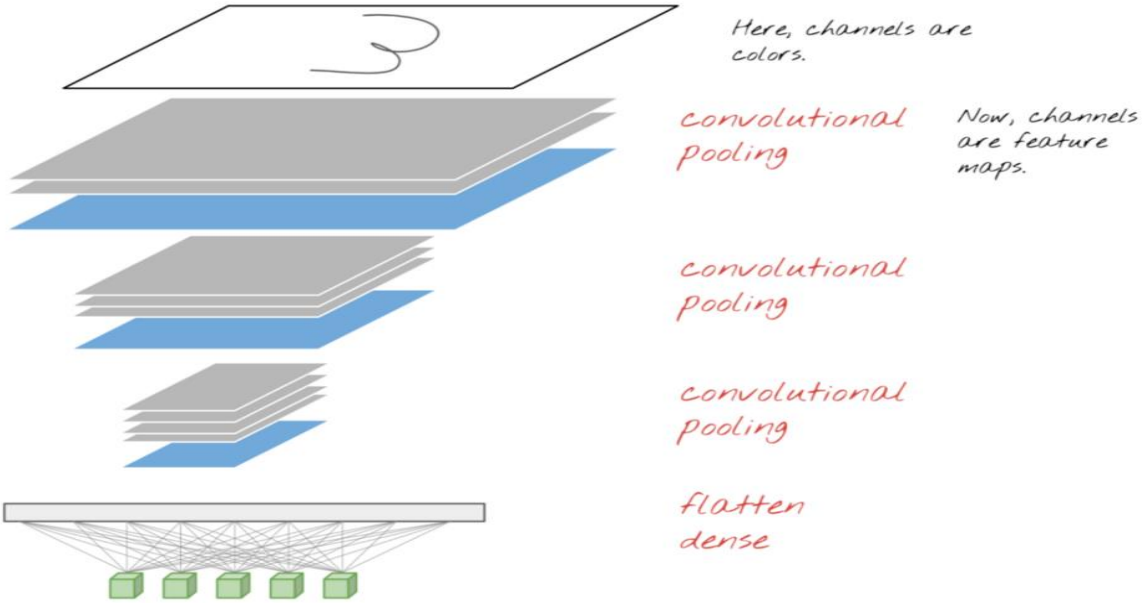
Πώς θα πρέπει να αλλάξετε τα βάρη ή τα ποσοστά εκμάθησης του νευρικού σας δικτύου για να μειώσετε τις απώλειες καθορίζεται από τους βελτιστοποιητές που χρησιμοποιείτε?Οι αλγόριθμοι βελτιστοποίησης είναι υπεύθυνοι[1] για τη μείωση των απωλειών και την παροχή των πιο ακριβών αποτελεσμάτων.



1

➢ ΣΥΝΗΘΕΙΣ ΑΡΧΙΤΕΚΤΟΝΙΚΗ

Προκειμένου να υλοποιηθούν τα CNN, η πιο επιτυχημένη αρχιτεκτονική χρησιμοποιεί μία ή περισσότερες στοίβες στρωμάτων convolution + pooling με ενεργοποίηση relu, ακολουθούμενη από flatten layer και στη συνέχεια dense layers.



Υλοποίηση Εργασίας

➢ Υλοποιήσεις:

Οι υλοποιήσεις περιλαμβάνουν ένα μη προ-εκπαιδευμένο δίκτυο και ένα προ-εκπαιδευμένο.

Μη προεκπαιδευμένο

Αρχικά έγινε προσπάθεια υλοποίησης της εργασίας με το μοντέλο VGG σαν βαση αλλά παρατηρήθηκε πως εκανε ευκολα overfit οπότε μείωσα τον αριθμό των layers και των φιλτρων και τα αποτελεσματα ήταν καλύτερα.

- Δημιουργία μοντέλου

```
# Create the model
model = models.Sequential()
model.add(layers.Conv2D(input_shape=(128,128,3),filters=32,kernel_size=(3,3),padding="same", activation="relu"))
model.add(layers.Conv2D(filters=64,kernel_size=(3,3),padding="same", activation="relu"))
model.add(layers.MaxPool2D(pool_size=(2,2),strides=(2,2)))
```

```
model.add(layers.Conv2D(filters=64, kernel_size=(3,3), padding="same", activation="relu"))
model.add(layers.Conv2D(filters=128, kernel_size=(5,5), padding="same", activation="relu"))
model.add(layers.MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(layers.Conv2D(filters=128, kernel_size=(7,7), padding="same", activation="relu"))
model.add(layers.Conv2D(filters=256, kernel_size=(7,7), padding="same", activation="relu"))
model.add(layers.MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(layers.Dropout(0.2))
model.add(layers.Flatten())
model.add(layers.Dense(units=256,activation="relu"))
model.add(layers.Dropout(0.4))
model.add(layers.Dense(units=128,activation="relu"))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(units=6, activation="softmax"))

# Show a summary of the model. Check the number of trainable parameters
model.summary()
```

Με output στοιχεία για τα layers μου.

Model: "sequential_14"

Layer (type)	Output Shape	Param #
conv2d_76 (Conv2D)	(None, 128, 128, 32)	896
conv2d_77 (Conv2D)	(None, 128, 128, 64)	18496
max_pooling2d_37 (MaxPooling	(None, 64, 64, 64)	0
conv2d_78 (Conv2D)	(None, 64, 64, 64)	36928
conv2d_79 (Conv2D)	(None, 64, 64, 128)	204928
max_pooling2d_38 (MaxPooling	(None, 32, 32, 128)	0
conv2d_80 (Conv2D)	(None, 32, 32, 128)	802944
conv2d_81 (Conv2D)	(None, 32, 32, 256)	1605888
max_pooling2d_39 (MaxPooling	(None, 16, 16, 256)	0
dropout_36 (Dropout)	(None, 16, 16, 256)	0
flatten_14 (Flatten)	(None, 65536)	0

```
_________________________________________________________________
dense_40 (Dense)             (None, 256)               16777472
_________________________________________________________________
dropout_37 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_41 (Dense)             (None, 128)               32896
_________________________________________________________________
dropout_38 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_42 (Dense)             (None, 6)                 774
=================================================================
Total params: 19,481,222
Trainable params: 19,481,222
Non-trainable params: 0
```

- Στη συνέχεια μέσω της ImageGenerator κανω scale down τις εικονες μου για να με διευκολύνει στους υπολογισμούς καθώς επισης και χρησιμοποιώ τεχνικές όπως το shear, το zoom, το horizontal flip και το rotate για να παράξω πολλές διαφορετικές εικόνες από το ίδιο dataset (το augmentation αναφέρθηκε αναλυτικότερα στο θεωρητικό υπόβαθρο).

```
train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.3,
                                   zoom_range=0.2,
                                   horizontal_flip=True,
                                   rotation_range=40,
                                  )
val_datagen  = ImageDataGenerator(rescale=1./255)
```

- Μετά μέσω της flow from directory δηλώνω το batch size μου δηλαδή το πόσες εικόνες θέλω να πάρω σε κάθε εποχή,το πως θελω να χωρίσω τις εικόνες μου, το τι μέγεθος θα εχουν και την επιλογή να κάνω shuffle τις εικόνες για να μην παίρνω κάθε φορά διαφορετικές.

```
train_generator = train_datagen.flow_from_directory(train_dir,
                                                    batch_size=240,
                                                    class_mode='categorical',
                                                    # color_mode='grayscale',
                                                    target_size=(128,128),
                                                    shuffle=True)
validation_generator =  val_datagen.flow_from_directory(validation_dir,
                                                        batch_size=240,
                                                        class_mode='categorical',
                                                        #  color_mode='grayscale',
```

```
target_size=(128,128))
```

- Στο επόμενο βήμα κάνω compile το μοντέλο δηλώνοντας αρχικά ποιον optimizer θέλω να χρησιμοποιήσω.Σε αυτό το βήμα χρησιμοποιήθηκαν και ο RMS και ο Adam και επιλέχθηκε ο RMS καθώς είχε καλύτερα αποτελέσματα. Επιπλέον δηλώνω μεσω του early stoping ότι θελω να σταματήσει να τρεχει το πρόγραμμα με βάση την ακρίβεια του validation test κάνωντας monitor την μέγιστη και σταματώντας το πρόγραμμα αν το val_acc δεν αυξηθεί μεσα σε 10 εποχές.Τέλος σηλώνω πως θέλω να αποθηκευτεί το μοντέλο με το καλύτερο val_acc έτσι ώστε να αποθηκέυσω το καλύτερο μου μοντέλο.

```
# Compile the model
model.compile(loss='categorical_crossentropy',
        optimizer=optimizers.RMSprop(lr=1e-4),
        metrics=['acc'])

earlyStopping = keras.callbacks.callbacks.EarlyStopping(monitor='val_acc', patience=10,
verbose=1, mode='max')
mcp_save = keras.callbacks.callbacks.ModelCheckpoint('small_last4.h5', save_best_only=True,
monitor='val_acc', mode='max',period=1,verbose=1)
```

- Τέλος κάνω train το μοντέλο μου δηλωνοντας σε ποιες εικόνες θα εκπαιδευτεί, ποια callbacks (από αυτά που εχουν αναφερθεί στο προηγούμενο βήμα θα χρησιμοποιήσει) καθώς επίσης και ποσες εποχές θα τρέξει. Για να έχω καλύτερη οπτική του μοντέλου μου καθώς και του πότε αρχίζει να κανει overfiting κανω στη συνέχεια plot τα δεδομένα μου και αποθήκευω στο τέλος το μοντέλο μου.

```
# Train the model
history = model.fit_generator(train_generator,
                callbacks=[earlyStopping, mcp_save],
                steps_per_epoch=train_generator.samples/train_generator.batch_size ,
                epochs=100,
                validation_data=validation_generator,

validation_steps=validation_generator.samples/validation_generator.batch_size,
                verbose=1)
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
```

```
plt.show()
# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
# Save the model
model.save('small_last4.h5')
```

❖ Το μοντέλο μου μου δίνει ακρίβεια 70% στα validation data και 64% στα test data.

Προεκπαιδευμένο

Δοκιμάστηκαν 3 προεκπαιδευμένα δίκτυα το VGG19, το VGG16 και το InceptionV3 με το VGG16 να μας δίνει τα καλύτερα αποτελέσματα.

- Αρχικά αφού φορτώσουμε το μοντέλο επιλέγουμε πόσα layers θα κανουμε freeze(δηλαδή πόσα layers δεν θα εκπαιδέυσουμε).Επιλέξαμε να εκπαιδεύσουμε 4 layers τρέχωντας το πρόγραμμα για διαφορετικές επιλογές αριθμού layers που μπορουμε να εκπαιδεύσουμε και βλέπωντας πως το 4 είναι η αποδοτικότερη επιλογή.

```
#Load the VGG model
vgg_conv = VGG16(weights='imagenet', include_top=False, input_shape=(332, 332, 3))
# Create the model
# Freeze the layers except the last 4 layers
for layer in vgg_conv.layers[:-4]:
    layer.trainable = False
```

- Στη συνέχεια χτίζουμε το μοντέλο μας συπληρώνωντας τα layers που θέλουμε καθώς και προσθέτωντας dropout για να αποφύγουμε overfitting

```
# Check the trainable status of the individual layers
for layer in vgg_conv.layers:
    print(layer, layer.trainable)
model = models.Sequential()
# Add the vgg convolutional base model
model.add(vgg_conv)
# Add new layers
model.add(layers.Flatten())
```

```
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(6, activation='softmax'))

# Show a summary of the model. Check the number of trainable parameters
model.summary()
```

- Στη συνέχεια μέσω της ImageGenerator κανω scale down τις εικονες μου για να με διευκολύνει στους υπολογισμούς .Εδώ δε χρειάστηκα μεγάλο batch size αφου το δίκτυο μου είναι ήδη εκπαιδευμένο στο Imagenet.

```
train_datagen = ImageDataGenerator(rescale=1./255)
val_datagen  = ImageDataGenerator(rescale=1./255)
```

- Μετά μέσω της flow from directory δηλώνω το batch size μου δηλαδή το πόσες εικόνες θέλω να πάρω σε κάθε εποχή,το πως θελω να χωρίσω τις εικόνες μου, το τι μέγεθος θα εχουν και την επιλογή να κάνω shuffle τις εικόνες για να μην παίρνω κάθε φορά διαφορετικές(shuffle).Επιλέγω size εικονών 332*332 καθώς ειχα καλύτερα αποτελέσματα από το 224*224.

```
train_generator = train_datagen.flow_from_directory(train_dir,
                                    batch_size=20,
                                    class_mode='categorical',
                                    #color_mode='grayscale',
                                    target_size=(332,332),
                                    shuffle=True)
validation_generator =  val_datagen.flow_from_directory(validation_dir,
                                     batch_size=20,
                                     class_mode='categorical',
                                    # color_mode='grayscale',
                                     target_size=(332,332))
```

❖ Το μοντέλο μου μου δίνει ακρίβεια 70% στα validation data και 64% στα test data.