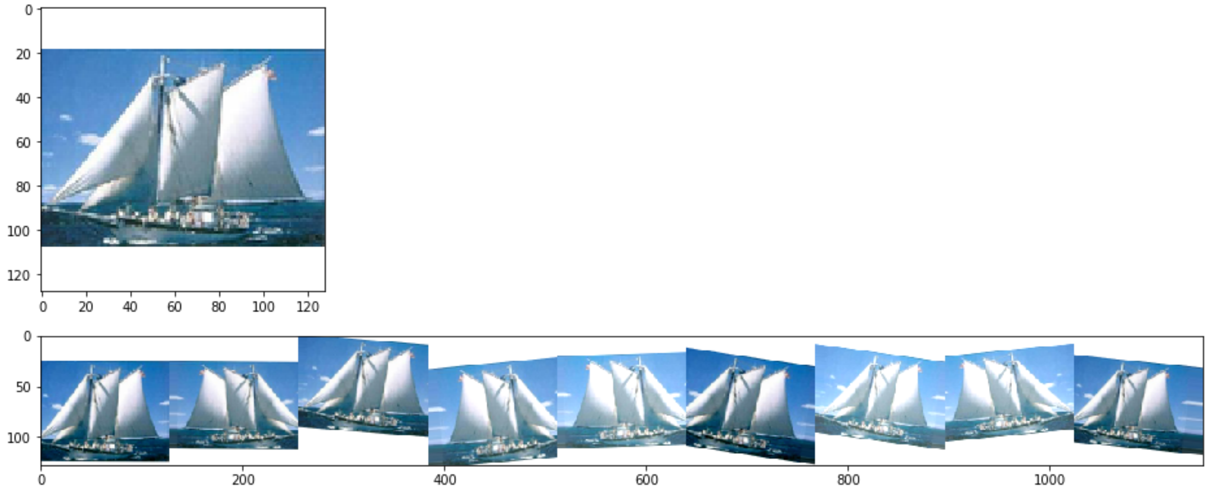
```
        break
plt.show()
concat_image = np.concatenate(concat_image, axis=1)
plt.figure(figsize=(16,16))
plt.imshow(concat_image)
plt.show()


```



5.7 Entrenamiento con Data Augmentation.

[inicio](#)

5.7.1 Preparamos los datos y el modelo

```
# crear subconjunto de datos
print("Creando conjunto de datos ...")
samples_per_class=100
data_subset, labels_subset = crear_subset(x_data, processed_y, samples_per_class=samples_per_class,number_of_classes=102,verb

# crear nuevos set de train, validate y test
print("Creando sets de train, validate y test ...")
X_train, y_train, X_validate, y_validate, X_test, y_test = crear_sets_tvt(data_subset=data_subset, labels_subset=labels_subse

# crea el modelo según la opción
print("Creando el modelo ...")
final_model_to_test = create_model_gec(opcion_modelo=4,optimizador="None")
final_model_to_test.compile(loss='categorical_crossentropy',
                            optimizer=Adam(lr=.0001),
                            metrics=['accuracy'])



```

5.7.2 Entrenamiento y evaluación del modelo

```
### entrenar utilizando el generador de data
# Compute quantities required for feature-wise normalization
# (std, mean, and principal components if ZCA whitening is applied).
#datagen.fit(x_train)

print("Entrenando ...")
start_time = timeit.default_timer()

# es necesario especificar el numero de steps cuando se usa un generator (para saber cuando dejar de generar datos en una epo
batch_size = 32
num_samples = X_train.shape[0] * 30 # multiplicar por 10 el numero de ejemplos

# Fit the model on the batches generated by datagen.flow().
history_augmentation = final_model_to_test.fit_generator(datagen.flow(X_train, y_train,batch_size=batch_size),
                            epochs=18,
                            validation_data=(X_test, y_test),
                            workers=4,
                            steps_per_epoch=num_samples//batch_size)

# guardamos los pesos
nombre_archivo_modelo = BASE_FOLDER + "modelo_0_final_1.hdf5"
final_model_to_test.save(nombre_archivo_modelo)
print("Modelo guardado en:", nombre_archivo_modelo)
```

```
print("Modelo guardado en:", nombre_archivo_modelo)

print("Tiempo de ejecución del experimento:",round(timeit.default_timer() - start_time,0))

plot_results_gec(history_augmentation)

print("\nEvaluando ...")
loss_val, acc_val = final_model_to_test.evaluate(X_validate, y_validate, verbose=0)
loss_test, acc_test = final_model_to_test.evaluate(X_test, y_test, verbose=0)
loss_all, acc_all = final_model_to_test.evaluate(x_data, y_cat, verbose=0)
print("Exactitud validate:",round((acc_val*100),4),"%, Loss:",round(loss_val,4))
print("Exactitud test:",round((acc_test*100),4),"%, Loss:",round(loss_test,4))
print("Exactitud todo el dataset:",round((acc_all*100),4),"%, Loss:",round(loss_all,4))

# epochs 9: 65%
# experimento batch 32, epochs 18, imagenes 30  74% validation
```

```
Epoch 1/18
6000/6000 [==============================] - 435s 73ms/step - loss: 2.2670 - acc: 0.4761 - val_loss: 1.5920 - val_acc: 0.61
Epoch 2/18
6000/6000 [==============================] - 412s 69ms/step - loss: 1.3154 - acc: 0.6533 - val_loss: 1.3522 - val_acc: 0.67
Epoch 3/18
6000/6000 [==============================] - 411s 69ms/step - loss: 0.9527 - acc: 0.7329 - val_loss: 1.2777 - val_acc: 0.70
Epoch 4/18
6000/6000 [==============================] - 408s 68ms/step - loss: 0.7354 - acc: 0.7873 - val_loss: 1.4373 - val_acc: 0.69
Epoch 5/18
6000/6000 [==============================] - 418s 70ms/step - loss: 0.5940 - acc: 0.8227 - val_loss: 1.3330 - val_acc: 0.72
Epoch 6/18
6000/6000 [==============================] - 406s 68ms/step - loss: 0.4896 - acc: 0.8515 - val_loss: 1.3749 - val_acc: 0.71
Epoch 7/18
6000/6000 [==============================] - 402s 67ms/step - loss: 0.4115 - acc: 0.8746 - val_loss: 1.3774 - val_acc: 0.71
Epoch 8/18
6000/6000 [==============================] - 412s 69ms/step - loss: 0.3641 - acc: 0.8881 - val_loss: 1.4022 - val_acc: 0.73
Epoch 9/18
6000/6000 [==============================] - 399s 66ms/step - loss: 0.3199 - acc: 0.9006 - val_loss: 1.4676 - val_acc: 0.73
Epoch 10/18
6000/6000 [==============================] - 407s 68ms/step - loss: 0.2955 - acc: 0.9087 - val_loss: 1.4016 - val_acc: 0.73
Epoch 11/18
6000/6000 [==============================] - 405s 67ms/step - loss: 0.2544 - acc: 0.9199 - val_loss: 1.5844 - val_acc: 0.72
Epoch 12/18
6000/6000 [==============================] - 406s 68ms/step - loss: 0.2453 - acc: 0.9234 - val_loss: 1.5697 - val_acc: 0.73
Epoch 13/18
6000/6000 [==============================] - 405s 68ms/step - loss: 0.2263 - acc: 0.9288 - val_loss: 1.4761 - val_acc: 0.73
Epoch 14/18
6000/6000 [==============================] - 409s 68ms/step - loss: 0.2113 - acc: 0.9335 - val_loss: 1.5127 - val_acc: 0.74
Epoch 15/18
6000/6000 [==============================] - 406s 68ms/step - loss: 0.1981 - acc: 0.9385 - val_loss: 1.5575 - val_acc: 0.73
Epoch 16/18
6000/6000 [==============================] - 404s 67ms/step - loss: 0.1911 - acc: 0.9396 - val_loss: 1.6264 - val_acc: 0.72
Epoch 17/18
6000/6000 [==============================] - 411s 69ms/step - loss: 0.1748 - acc: 0.9445 - val_loss: 1.7610 - val_acc: 0.72
Epoch 18/18
6000/6000 [==============================] - 422s 70ms/step - loss: 0.1680 - acc: 0.9466 - val_loss: 1.5964 - val_acc: 0.74
```

```
# vemos los resultados   # 61%   best test 66%
plot_results(history_augmentation)

# predecimos para verificar la exactitud conseguida
#predict_sample(X_test,y_test,model=test_model)
print("Exactitud en subconjunto de test:")
predict_accuracy(X_test, y_test, model,verbose=True)

print("\nExactitud en todo el dataset:")
predict_accuracy(x_data, y_cat, model,verbose=True)
```