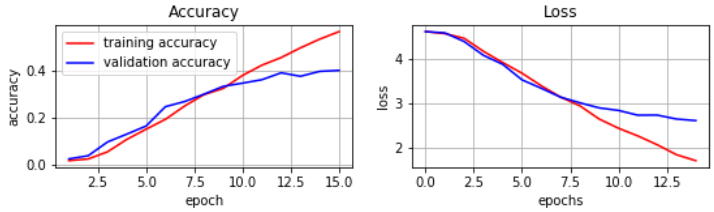
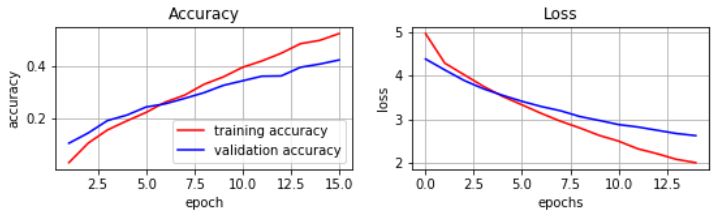
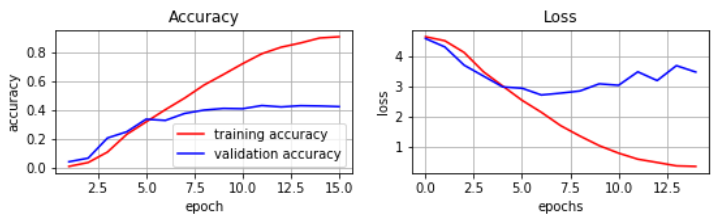
```
================================================================================
exp = Experimento_1b_1_3
modelo = 3
samples_per_class = 100
opti = Adam
batch = 128
epochs = 15
run = True
----------------------------------------------------------
Creando juego de datos ...
Entrenando ...
Tiempo de entrenamiento: ** 66.0 **
```



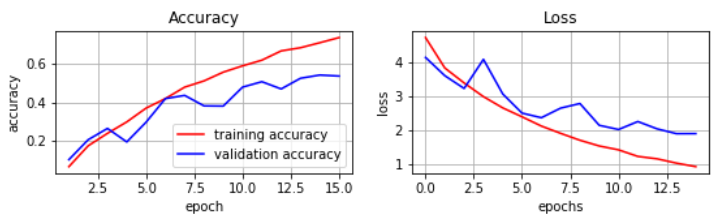

```
Evaluando ...
Exactitud validate: 40.0 %, Test: 43.33 %
Tiempo de ejecución del experimento: 67.0
================================================================================
exp = Experimento_1b_1_4
modelo = 4
samples_per_class = 100
opti = Adam
batch = 128
epochs = 15
run = True
----------------------------------------------------------
Entrenando ...
Tiempo de entrenamiento: ** 40.0 **
```




```
Evaluando ...
Exactitud validate: 42.4 %, Test: 43.44 %
Tiempo de ejecución del experimento: 41.0
================================================================================
exp = Experimento_1b_2_3
modelo = 3
samples_per_class = 100
opti = rmsprop
batch = 128
epochs = 15
run = True
----------------------------------------------------------
Entrenando ...
Tiempo de entrenamiento: ** 66.0 **
```




```
Evaluando ...
Exactitud validate: 42.4 %, Test: 45.0 %
Tiempo de ejecución del experimento: 67.0
================================================================================
exp = Experimento_1b_2_4
modelo = 4
samples_per_class = 100
opti = rmsprop
batch = 128
epochs = 15
run = True
----------------------------------------------------------
Entrenando ...
Tiempo de entrenamiento: ** 42.0 **
```



```
Evaluando ...
Exactitud validate: 53.54 %, Test: 55.42 %
Tiempo de ejecución del experimento: 43.0
Tiempo de ejecución todo el batch: 261.0
```

Resultados Lote 1a

Experimento	Muestras*Clase	Optimizador	Modelo	Batch Size	epocas	Tiempo Entrenamiento	Exac. val.	Exact. test
Experimento_1b_1_3	100	Adam	3	128	15	66	40.00%	43.33%
Experimento_1b_1_4	100	Adam	4	128	15	40	42.40%	43.44%
Experimento_1b_2_3	100	rmsprop	3	128	15	66	42.40%	45.00%
Experimento_1b_2_4	100	rmsprop	4	128	15	42	53.54%	**55.42%**

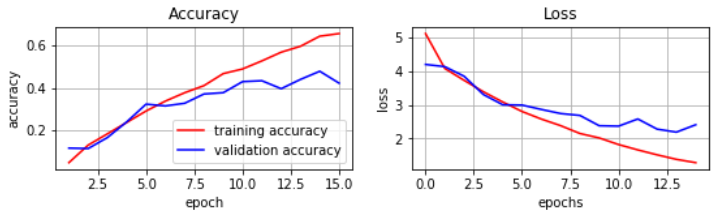
Si comparamos con los resultados anteriores podemos observar que el **batch_size=32** funciona mejor en la mayoria de los casos excepto en el **Experimento_1b_2_4**.

Experimento	Muestras*Clase	Optimizador	Modelo	Batch Size	epocas	Tiempo Entrenamiento	Exac. val.	Exact. test
Experimento_1_3	100	Adam	3	32	15	73	44.38%	48.44%
Experimento_1_4	100	Adam	**4**	32	15	44	50.10%	**52.08%**
Experimento_2_3	100	rmsprop	**3**	32	15	71	47.71%	**53.44%**
Experimento_2_4	100	rmsprop	4	32	15	43	48.12%	52.60%

```
# definimos los experimentos del lote 1
number_of_classes = 102
experimentos = [
                {"exp":"Experimento_1b_2_4","modelo":4,"samples_per_class":100,"opti":"rmsprop","batch":2
                ]

start_time = timeit.default_timer()
hacer_experimento_gec(experimentos)
print("Tiempo de ejecución todo el batch:",round(timeit.default_timer() - start_time,0))
```

```
================================================================================
exp = Experimento_1b_2_4
modelo = 4
samples_per_class = 100
opti = rmsprop
batch = 256
epochs = 15
run = True
------------------------------------------------------------
Creando juego de datos ...
Entrenando ...
Tiempo de entrenamiento: ** 40.0 **
```




```
Evaluando ...
Exactitud validate: 42.29 %, Test: 45.42 %
Tiempo de ejecución del experimento: 41.0
Tiempo de ejecución todo el batch: 76.0
```

```
1
```

```
1
```

```
model = models.Sequential()
print("Input dimensions: ",X_train.shape[1:])

#model.add(layers.Conv2D(32, (3, 3), input_shape=X_train.shape[1:]))
model.add(layers.Conv2D(32, (3, 3),use_bias=False, input_shape=X_train.shape[1:]))
model.add(layers.BatchNormalization())  ########
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))

#model.add(layers.Conv2D(32, (3, 3)))
model.add(layers.Conv2D(32, (3, 3),use_bias=False))
model.add(layers.BatchNormalization())  ########
model.add(layers.Activation('relu'))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))

```