UNIVERSIDAD
POLITÉCNICA
DE MADRID

Práctica 4: Clasificación de imágenes

Cesar García Cabeza
Enol García González

Curso 2020-21

Fecha: Enero de 2021

Asignatura:
Visión por Computador

Profesor:
D. Luis Baumela Molina

Índice

1. Clasificación de imágenes

El problema al que nos enfrentamos en este práctica es el de clasificación de imágenes usando técnicas de aprendizaje profundo. Éste es uno de los problemas más comunes dentro del campo de la visión por computador moderno.

En las imágenes aparecerá únicamente un objeto perteneciente a una clase y nuestra tarea será la de clasificar dicha imagen como la categoría del objeto. No debemos confundir este problema con el de detección de objetos, en el cual en una misma imagen aparecen varios objetos y la tarea es la de reconocerlos todos a la vez.

Para resolver este problema haremos uso de redes neuronales: primero usando redes neuronales *feedforward* y después usando redes neuronales con capas convolucionales.

2. Datasets

Para el desarrollo de esta práctica haremos uso de dos *datasets* de imágenes conocidos: Cifar-10 y *German Traffic Sign Recognition*.

Cifar-10 Este dataset está formado por 60000 imágenes a color de 32x32 divididas en 10 clases, con 6000 imágenes de cada clase. Las categorías a las que pertenecen las imágenes son: avión, coche, pájaro, gato, ciervo, perro, rana, caballo, barco y camión.

German Traffic Sign Este dataset está formado por 40000 imágenes a color correspondientes a 43 clases distintas de señales de tráfico. Algunos de los tipos de señales de tráfico que aparecen son: límites de velocidad, stop, no adelantar, etc.

Uno de los problemas de este dataset es que está desbalanceado, teniendo más ejemplos de determinadas clases y menos de otras.

3. Metodología

En esta sección explicaremos las técnicas empleadas en la implementación de nuestras redes. En concreto, se han aplicado técnica de *regularization* y *data augmentation*. Además, explicaremos los dos tipos de redes usados: redes *feed forward* densamente conectadas y redes con capas convolucionales.

3.1. *Regularization*

Las técnicas de regularización permiten reducir el *overfitting* de las redes haciendo así más fácil el proceso de construcción de las mismas. Si recordamos, el *overfitting* es un suceso que ocurre cuando nuestra red es demasiado compleja para el problema que estamos tratando y se ajusta demasiado bien a los datos. Por lo tanto, estas técnicas de regularización lo que nos permiten es reducir de diversas formas la complejidad de nuestras redes, para así, reducir el *overfitting*.

Dentro de las técnicas de regularización, hemos decidido usar dos muy conocidas: *Dropout* y *Batch Normalization*.

3.1.1. *Dropout*

La primera técnica empleada es *Dropout* [4]. Esta técnica es muy sencilla y consiste en que, con una cierta probabilidad P, cada neurona se desactiva y es como si no existiese. Esta técnica se puede aplicar a todas las capas que queramos, incluso con distinta probabilidad para cada capa; pero solamente se aplica durante el entrenamiento de nuestras redes, nunca durante la predicción.

Un ejemplo de esta técnica lo podemos observar en la Figura 1. En la imagen de la izquierda, podemos comprobar cómo la red está formada por una serie de neuronas siempre activas. Sin embargo, en la derecha podemos ver cómo se le ha aplicado *dropout* a todas las capas de la red, omitiéndose en la primera capa 2 neuronas, en la segunda 3 y en la tercera de nuevo 2. Al omitir estas neuronas, observamos como el número de conexiones de nuestra red se ve disminuido y por tanto la complejidad.

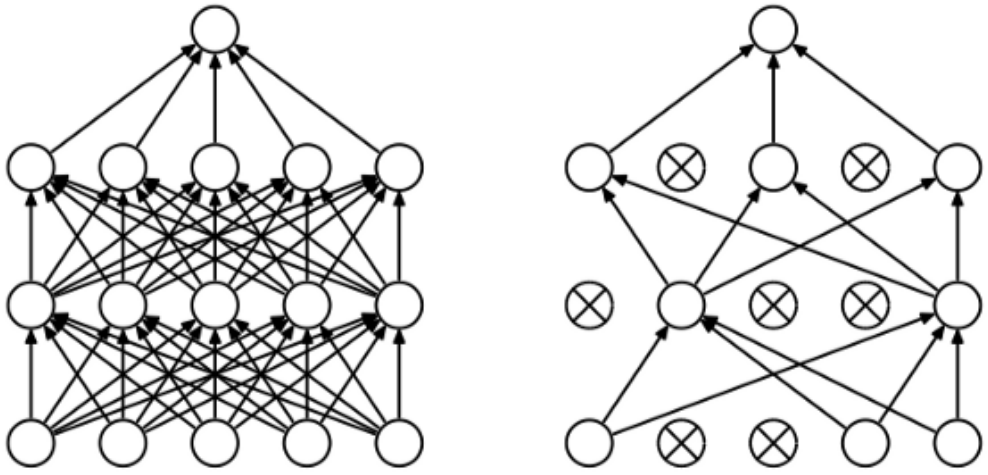

Figura 1: Ejemplo de aplicación de *Dropout*.

Fuente: http://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf

3.1.2. *Batch Normalization*

Batch Normalization [3] es un técnica de estandarización que se aplica a cada neurona en cada capa en cada *batch*. Esto se consigue calculando la media y la desviación estándar de cada capa, y normalizando las neuronas con esos valores. Esta técnica se puede aplicar a las neuronas antes o después de la activación; sin embargo, lo común es aplicarlo antes de la activación [3].

Además, usando esta técnica se consigue reducir el tiempo de entrenamiento de la red.

3.2. *Data augmentation*

Uno de los problemas del uso de técnicas de *deep learning* es la necesidad de datasets muy grandes, con un gran número de ejemplos. A veces, para determinadas tareas en imposible conseguir un dataset del tamaño requerido para ese problema concreto. Es ahí donde aparece el uso de la técnica *data augmentation*. Esta técnica nos permite aumentar el tamaño de nuestro dataset mediante pequeñas modificaciones de los datos originales.

En nuestro caso, nuestros datasets están formados por imágenes. Las técnicas qué hemos aplicado han sido las siguientes:

- ***Random zoom***: aplica un factor de zoom aleatoriamente a las imágenes tanto en el eje vertical como en el horizontal. Esto se aplica durante el entrenamiento y la predicción.
- ***Random rotation***: rota aleatoriamente las imágenes en un determinado rango. Esta técnica se suele aplicar únicamente durante el entrenamiento.
- ***Random flip***: da la vuelta aleatoriamente a las imágenes tanto vertical como horizontalmente. En nuestro caso, hemos decidido dar la vuelta únicamente horizontalmente.

Podemos comprobar, como con apenas estas 3 sencillas técnicas, conseguimos aumentar el tamaño de nuestro dataset aportando mayor variedad.

3.3. Tipos de redes

En esta práctica haremos uso de dos tipos de redes neuronales para resolver nuestros dos problemas de clasificación y comprobar como el uso de redes con capas convolucionales mejora el rendimiento.

3.3.1. Red *feed forward* densamente conectada

El primer tipo de red empleado es una *dense feed forward*. Este tipo de red corresponde con los modelos clásicos, en los que todas las neuronas de una capa están conectadas con todas las neuronas de la capa siguiente. Este tipo de modelo nos permite configurarlo de infinitas formas distintas, pudiendo jugar con el número de capas de neuronas y el número de neuronas por capas.

Al tratarse de un problema de clasificación, podremos jugar con cualquier combinación de capas y neuronas pero siempre la última capa estará formada por el número de clases de nuestro de dataset a la que aplicaremos la función *softmax*.

3.3.2. Red con capas convolucionales

Este tipo de modelo de red neuronal fue introducido a finales de los 80 pero no fue hasta el 2012 cuando empezó a cobrar fuerza entre la comunidad del aprendizaje profundo. Este tipo de red está formada por capas con filtros convolucionales de una o dos dimensiones que permiten extraer características abstractas de las imágenes. A la hora de configurar nuestra red podremos jugar con distintos parámetros como el tamaño del kernel, el número de filtros, si se aplica *padding* o no...

En nuestro problema de clasificación, nuestras redes estarán formadas por dos partes: la primera, compuesta por capas convolucionales que serán las encargadas de extraer las características de las imágenes y la segunda, compuesta por capas de neuronas densamente conectadas encargadas de clasificar las imágenes en base a las características extraídas.

4. Arquitecturas

En esta sección se explicarán las arquitecturas probadas para ambos problemas de clasificación. Se emplearon las mismas arquitecturas para ambos datasets para comprobar así su funcionamiento independientemente del dataset.

En cuanto a las técnicas de regularización explicadas anteriormente -*batch normalization* y *dropout*- no se recomienda usarlas a la vez [2] [3]. De todas formas, para cada arquitectura vamos a probar por separado cada técnica y una versión en la que usamos las dos a la vez.

4.1. Redes *feed forward* densamente conectadas

Se han probado tres arquitecturas distintas, empezando de más simple a más compleja. En las tres arquitecturas se introdujeron al inicio capas correspondientes a los tres tipos de *data augmentation* explicados en la Sección 3.2. Además, se probó cada arquitectura con *dropout*, con *batch normalization* y con los dos a la vez. Por lo tanto, tendríamos 9 redes distintas.

A continuación, ilustraremos las tres arquitecturas utilizadas sin incluir *dropout* ni *batch normalization*, indicándose esto en la sección de resultados. Destacar que las tres compartirán una capa de salida de 10 neuronas (correspondientes a las 10 clases de objetos en CIFAR) y una capa de aplanamiento al principio generando de una imagen de $32 \times 32 \times 3$ una capa de 3072 neuronas.

- En la Figura 2[1] podemos ver la arquitectura más simple de todas formada únicamente por una capa oculta de 128 neuronas.
- La siguiente arquitectura (Figura 3) ya tiene 4 capas ocultas de 128, 64, 32 y 16 neuronas respectivamente.
- Finalmente, la tercera arquitectura (Figura 4) está nuevamente formada por 4 capas ocultas pero aumentando el número de neuronas de cada una de ellas a 1024, 512, 64 y 64 respectivamente.

Para el dataset de las señales de tráfico se han utilizado las mismas redes pero cambiando la capa final de 10 neuronas por otra de 43, correspondiente al número de clases de este dataset. Las imágenes de las estructuras para este dataset se pueden ver en el Anexo A.

[1] Para realizar estos diagramas se ha utilizado el código de https://github.com/gwding/draw_convnet.

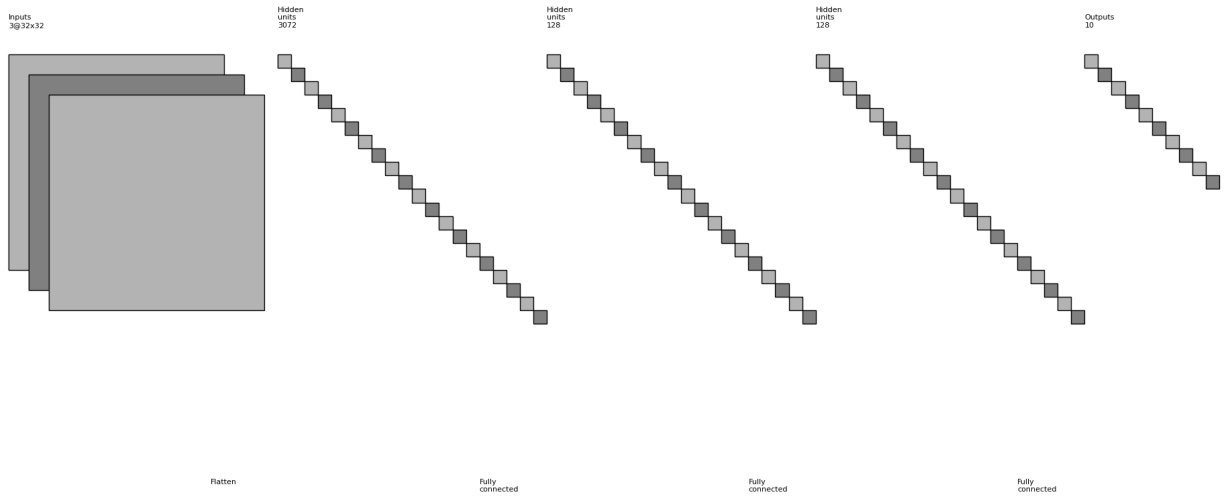

Figura 2: Arquitecturas: modelo 1 CIFAR

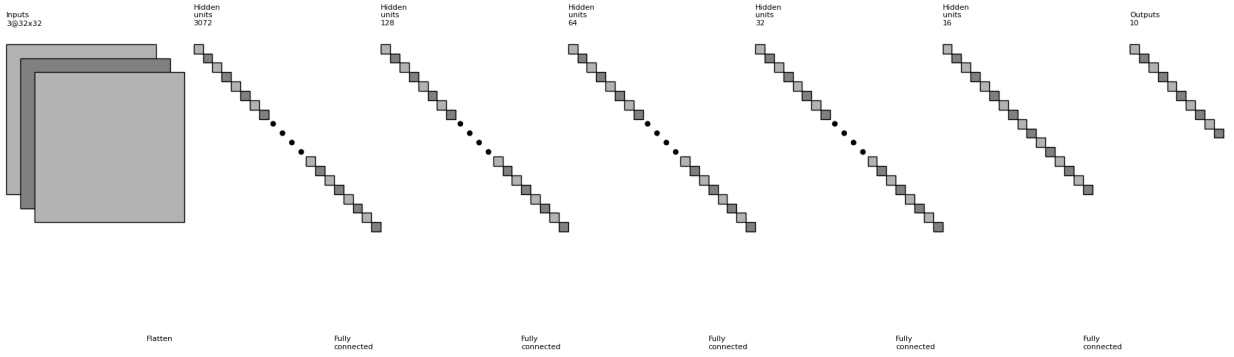

Figura 3: Arquitecturas: modelo 2 CIFAR

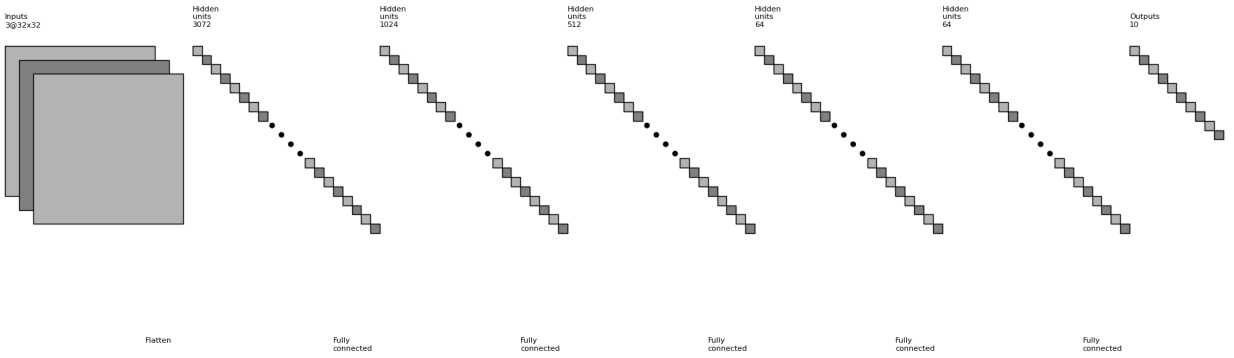

Figura 4: Arquitecturas: modelo 3 CIFAR

4.2. Redes convolucionales

Se han probado dos redes convolucionales para el primer dataset, Cifar-10, y para el segundo dataset se han probado la mismas redes convolucionales y la sugerida en CITAR. Tras una breve experimentación se comprobó que en estos casos incluir las capas de *data augmentation* no era buena idea provocando resultados muy malos tanto en entrenamiento como en validación.

Al igual que con las redes densamente conectadas, se ha probado el uso de *dropout*, *batch normalization* y las dos a la vez. Sin embargo, a la hora de mostrar las arquitecturas haremos igual que en el apartado anterior omitiendo estas características que se detallarán en el apartado de resultados.

Para simplificar la explicación, llamaremos *capa* (en cursiva) a lo siguiente:

1. Dos capa convolucionales de k filtros con un kernel de 3x3, padding *same* y stride 1. De esta forma el tamaño de salida coincidirá con el de entrada.

2. Una capa de *Max-Pooling* con un kernel de 2x2 y con un stride de 2. De esta forma el tamaño de salida será la mitad que el de entrada.

A continuación mostremos los modelos para el dataset CIFAR, pudiéndose encontrar los mismos modelos pero para el dataset de las señales en el Anexo B.

4.2.1. Modelo 1

Una vez explicado nuestro concepto de *capa*, la estructura (Figura 5) de la primera red convolucional probada es la siguiente:

- Una imagen de 32x32x3 de entrada.
- Una *capa* de 32 filtros.
- Una *capa* de 64 filtros.
- Una *capa* de 128 filtros.
- Una capa de aplanamiento.
- Una capa de salida con el mismo número de neuronas que clases tiene el dataset.

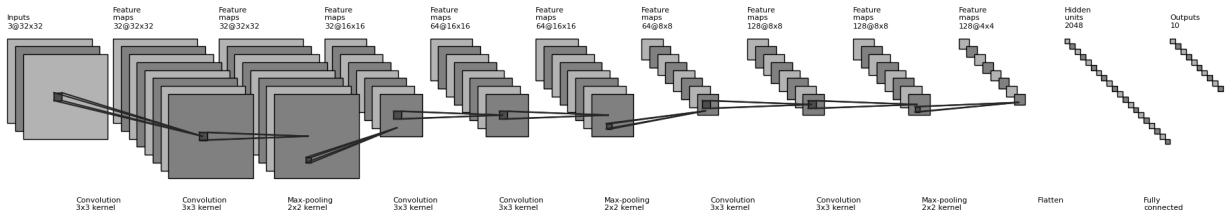

Figura 5: Arquitecturas: convolucional simple CIFAR

4.2.2. Modelo 2

La estructura (Figura 6) de la segunda red convolucional probada es la siguiente:

- Una imagen de 32x32x3 de entrada.
- Una *capa* de 32 filtros.
- Una *capa* de 64 filtros.
- Una *capa* de 128 filtros.

- Una capa de aplanamiento.
- Una capa densamente conectada del mismo tamaño que la anterior (2048).
- Una capa de salida con el mismo número de neuronas que clases tiene el dataset.

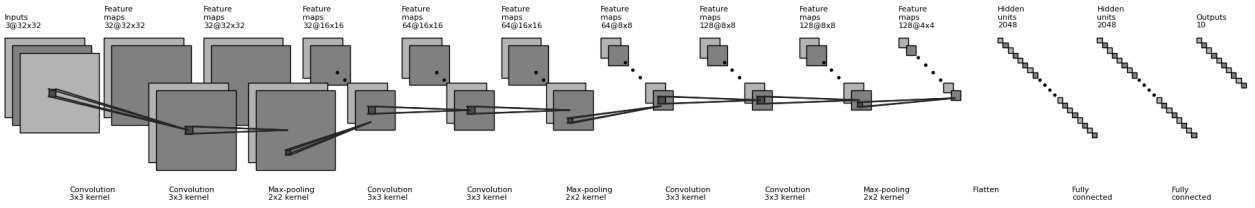

Figura 6: Arquitecturas: convolucional doble CIFAR

4.2.3. Artículo

Además, hemos decidido probar la red propuesta en [1]. La estructura de esta red se puede observar en la Figura 7.

Layer	Type	# Maps & neurons	Kernel
0	Input	3 maps of 48 × 48 neurons	
1	Convolutional	100 maps of 42 × 42 neurons	7 × 7
2	Max pooling	100 maps of 21 × 21 neurons	2 × 2
3	Convolutional	150 maps of 18 × 18 neurons	4 × 4
4	Max pooling	150 maps of 9 × 9 neurons	2 × 2
5	Convolutional	250 maps of 6 × 6 neurons	4 × 4
6	Max pooling	250 maps of 3 × 3 neurons	2 × 2
7	Fully connected	300 neurons	1 × 1
8	Fully connected	43 neurons	1 × 1

Figura 7: Arquitectura propuesta en el artículo

Fuente: [1]

5. Experimentos

5.1. CIFAR-10

5.1.1. *Dense feed forward*

En la Tabla 1 se muestran los resultados de los experimentos con redes neuronales densamente conectadas para el primer dataset. Se han entrenado las redes durante 50 épocas, se ha usado ReLu como función de activación, un *learning rate* de 0,001 y se han incluido al principio las capas de data augmentation explicadas anteriormente.

Las gráficas del entrenamiento de cada modelo se pueden encontrar en el Anexo C nombradas por cada ID usado en la tabla anterior.

5.1.2. Convolucionales

A continuación, en la Tabla 2 se muestran los resultados de usar redes convolucionales. Se han entrenado durante 20 épocas, se ha usado ReLu como función de activación, un *learning rate* de 0,001, Adam como algoritmo de optimización y **no** se han incluido capas de *data augmentation*.

En este caso, las posibles configuraciones son:

- ***Dropout***: sí o no.

Tabla 1: Cifar-10: experimentos redes densamente conectadas

ID	Arquitectura	*Dropout*	*Batch Normalization*	Training time(s)	Acc train	Acc val	Acc test
0	Modelo 1	Sí	Sí	232,05	44,43 %	43,78 %	43,78 %
1	Modelo 1	Sí	No	213,22	38,14 %	43,53 %	43,53 %
2	**Modelo 1**	**No**	**Sí**	**211,69**	**53,57 %**	**50,58 %**	**50,58 %**
3	Modelo 2	Sí	Sí	226,78	37,58 %	44,11 %	44,11 %
4	Modelo 2	Sí	No	210,74	24,38 %	30,57 %	30,57 %
5	Modelo 2	No	Sí	215,06	50,88 %	50,05 %	50,05 %
6	Modelo 3	Sí	Sí	651,92	49,44 %	48,74 %	48,74 %
7	Modelo 3	Sí	No	586,01	37,11 %	44,09 %	44,09 %
8	Modelo 3	No	Sí	556,12	60,91 %	61,27 %	51,27 %

Tabla 2: Cifar-10: experimentos redes convolucionales

ID	*Dropout*	*Batch Normalization*	*Capa extra*	Training time (s)	Acc train	Acc val	Acc test
9	Sí	Después	No	4681,24	87,10 %	80,28 %	80,27 %
10	**Sí**	**Antes**	**No**	**5196,81**	**88,23 %**	**81,80 %**	**81,80 %**
11	Sí	No	No	2428,5	81,79 %	79,60 %	79,60 %
12	No	Después	No	4637,94	100,00 %	79,82 %	79,82 %
13	No	Antes	No	4822,82	97,64 %	75,81 %	75,81 %
14	Sí	Después	Sí	4287,04	79,71 %	73,80 %	73,80 %
15	Sí	Antes	Sí	3084,44	81,53 %	78,74 %	78,74 %
16	Sí	No	Sí	1588,19	82,21 %	79,99 %	79,99 %
17	No	Después	Sí	3025,16	94,72 %	75,28 %	75,27 %
18	No	Antes	Sí	2935,26	95,39 %	71,99 %	71,99 %

- ***Batch Normalization***: no, antes o después de la activación.
- **Capa extra**: sí o no, dependiendo de si se incluye otra capa densamente conectada al final. Si hablamos de capa extra sí correspondería con el modelo 2 y si no tiene capa extra sería el modelo 1 (Sección 4.2)

Las gráficas del entrenamiento de cada modelo se pueden encontrar en el Anexo C nombradas por cada ID usado en la tabla anterior.

5.2. Traffic Sign

5.2.1. *Dense feed forward*

En la Tabla 3 se muestran los experimentos para el dataset de señales de tráfico usando redes neuronales densamente conectadas. Todas las ejecuciones han sido durante 150 épocas, usando Adam como algoritmo de optimización, un *learning rate* de 0.001, ReLu como función de activación y se han incluido capas de *data augmentation*.

Tabla 3: Traffic Sign: experimentos redes densamente conectadas

ID	Arquitectura	*Dropout*	*Batch Normalization*	Tiempo de entrenamiento (s)	Acc train	Acc val	Acc test
19	Modelo 1	Sí	Sí	300,00	59,49 %	50,00 %	54,01 %
20	Modelo 1	Sí	No	308,36	45,58 %	50,79 %	54,29 %
21	**Modelo 1**	**No**	**Sí**	**309,45**	**84,80 %**	**55,95 %**	**59,83 %**
22	Modelo 2	Sí	Sí	343,63	31,28 %	35,32 %	35,73 %
23	Modelo 2	Sí	No	331,67	24,08 %	30,02 %	31,57 %
24	Modelo 2	No	Sí	327,96	72,74 %	46,43 %	49,31 %
25	Modelo 3	Sí	Sí	713,40	50,98 %	41,27 %	45,43 %
26	Modelo 3	Sí	No	687,99	46,13 %	42,86 %	47,64 %
27	Modelo 3	No	Sí	670,85	91,49 %	48,41 %	49,31 %

Las gráficas del entrenamiento se pueden encontrar en el Anexo D nombradas por el ID de cada experimento.

5.2.2. Convolucionales

En la Tabla 4 se muestran los experimentos con las mismas dos redes convolucionales que se usaron para el dataset de Cifar.

Todas las ejecuciones han sido durante 25 épocas, usando Adam como algoritmo de optimización, un *learning rate* de 0.001, ReLu como función de activación y **no** se han incluido capas de *data augmentation*.

Tabla 4: Traffic Sign: experimentos redes convolucionales

ID	*Dropout*	*Batch Normalization*	*Capa extra*	Training time (s)	Acc train	Acc val	Acc test
28	Sí	Después	No	3729,84	68,42 %	19,84 %	19,11 %
29	Sí	Antes	No	3754,77	98,78 %	20,63 %	23,82 %
30	Sí	No	No	1774,84	98,71 %	71,03 %	70,63 %
31	No	Después	No	3674,45	86,42 %	33,33 %	36,56 %
32	No	Antes	No	3258,85	97,35 %	25,40 %	27,42 %
33	Sí	Después	Sí	2613,29	55,88 %	4,76 %	2,49 %
34	Sí	Antes	Sí	2503,4	95,10 %	8,73 %	9,69 %
35	**Sí**	**No**	**Sí**	**1349,29**	**92,80 %**	**74,21 %**	**76,17 %**
36	No	Después	Sí	2508,07	75,65 %	11,51 %	18,83 %
37	No	Antes	Sí	2435,33	98,27 %	26,19 %	37,39 %

Las gráficas del entrenamiento se pueden encontrar en el Anexo D nombradas por el ID de cada experimento.

5.2.3. Red artículo

En la Tabla 5 se muestran las dos ejecuciones de la red convolucional explicada en la sección 4.2.3.

Todas las ejecuciones han sido durante 25 épocas, usando Adam como algoritmo de optimización, un *learning rate* de 0.001, Tanh como función de activación y se han incluido capas de *data augmentation*.

Tabla 5: Traffic Sign: experimentos red convolucional artículo

ID	*Dropout*	Training time (s)	Acc train	Acc val	Acc test
38	Sí	516,62	77,17 %	68,65 %	72,57 %
39	**No**	**517,58**	**87,67 %**	**69,44 %**	**75,90 %**

Las gráficas del entrenamiento se pueden encontrar en el Anexo D nombradas por el ID de cada experimento.

6. Conclusiones

Cifar-10 Para este dataset obtenemos un 50,58 % de exactitud usando redes densamente conectadas y conseguimos mejorar este número usando redes convolucionales hasta un 81,80 %.

Traffic Sign Obtenemos un 59,83 % de exactitud usando únicamente redes densamente conectadas y conseguimos mejorar el rendimiento usando redes convolucionales hasta un 76,17 %. Además, la red propuesta en [1] no mejora (entrenándola durante 25 épocas) el rendimiento de nuestra mejor red convolucional.

También se puede comprobar de manera generalizada que la combinación que mejor funciona para las redes densamente conectadas es usar *batch normalization* sin *dropout*.

Referencias

[1] D. Cireşan, U. Meier, J. Masci y J. Schmidhuber, "Multi-column deep neural network for traffic sign classification," *Neural Networks*, vol. 32, págs. 333-338, 2012, ISSN: 08936080. DOI: 10.1016/j.neunet.2012.02.023. dirección: http://dx.doi.org/10.1016/j.neunet.2012.02.023.

[2] I. Goodfellow, Y. Bengio, A. Courville e Y. Bengio, *Deep learning*, 2. MIT press Cambridge, 2016, vol. 1.

[3] S. Ioffe y C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *CoRR*, vol. abs/1502.03167, 2015. arXiv: 1502.03167. dirección: http://arxiv.org/abs/1502.03167.

[4] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever y R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, n.º 56, págs. 1929-1958, 2014. dirección: http://jmlr.org/papers/v15/srivastava14a.html.

A. Arquitecturas *dense feed forward* Traffic Sign

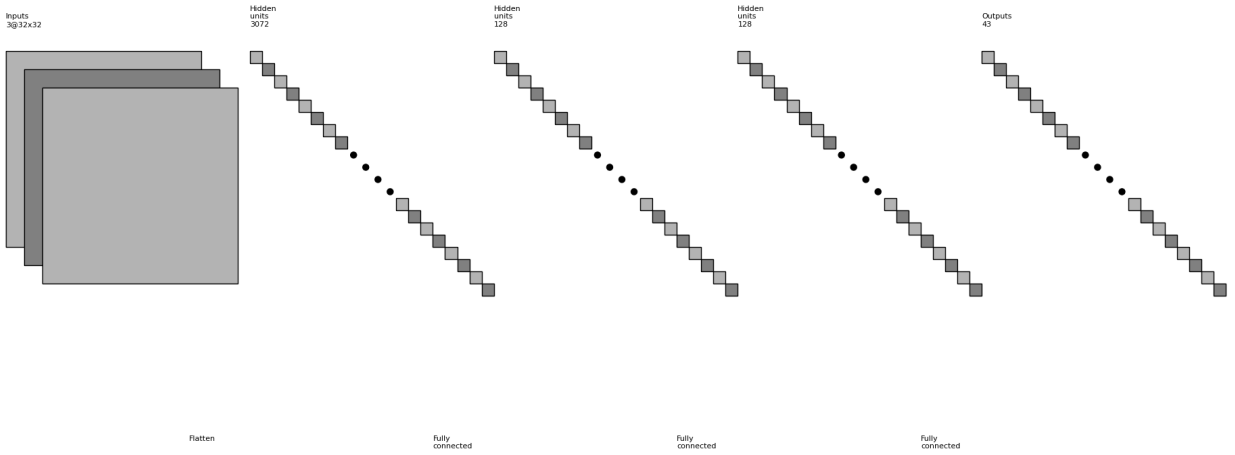

Figura 8: Arquitecturas: modelo 1 Traffic

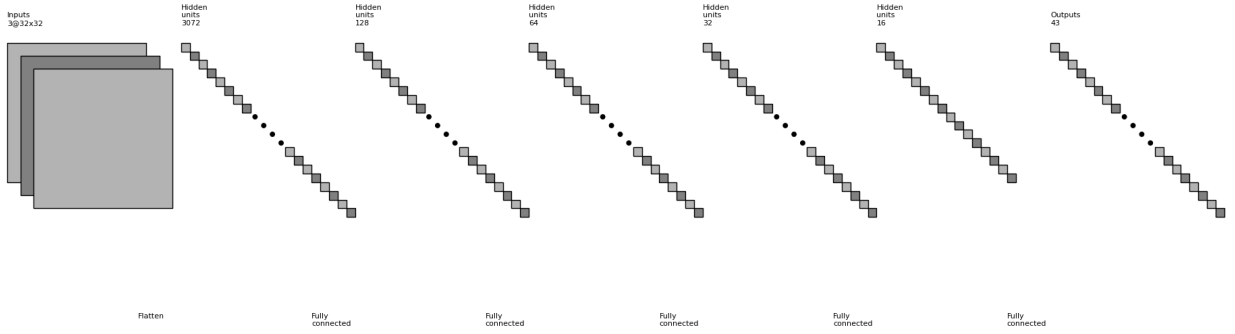

Figura 9: Arquitecturas: modelo 2 Traffic

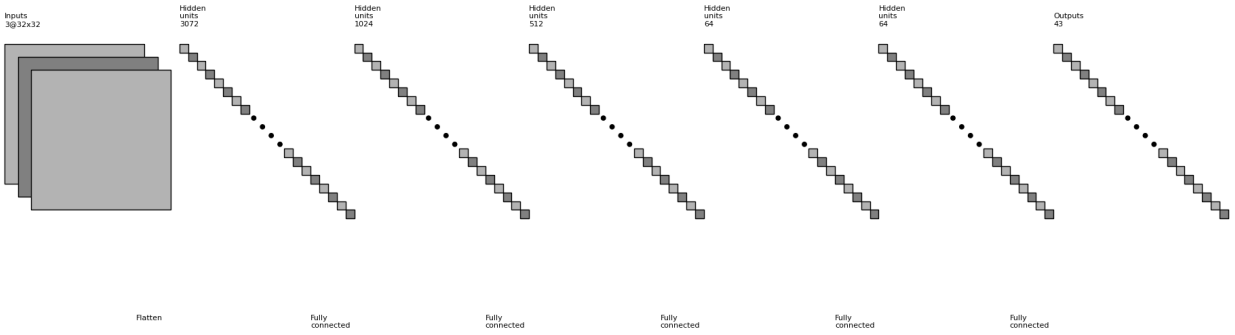

Figura 10: Arquitecturas: modelo 3 Traffic

B. Arquitecturas *convolucionales* Traffic Sign

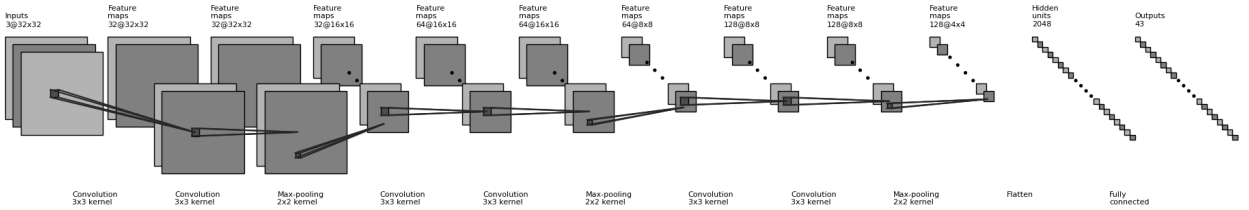

Figura 11: Arquitecturas: convolucional simple Traffic

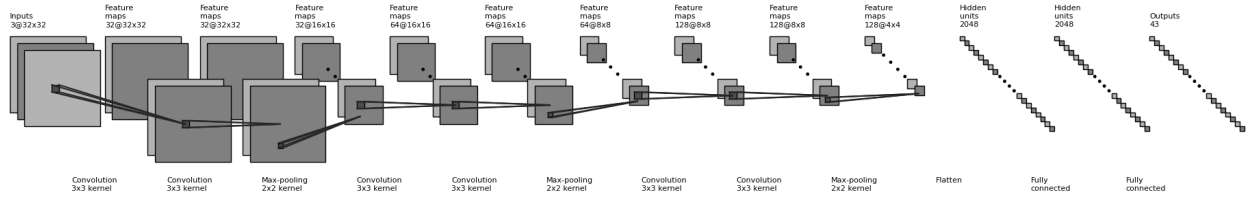

Figura 12: Arquitecturas: convolucional doble Traffic

C. Gráficas de entrenamiento: Cifar-10

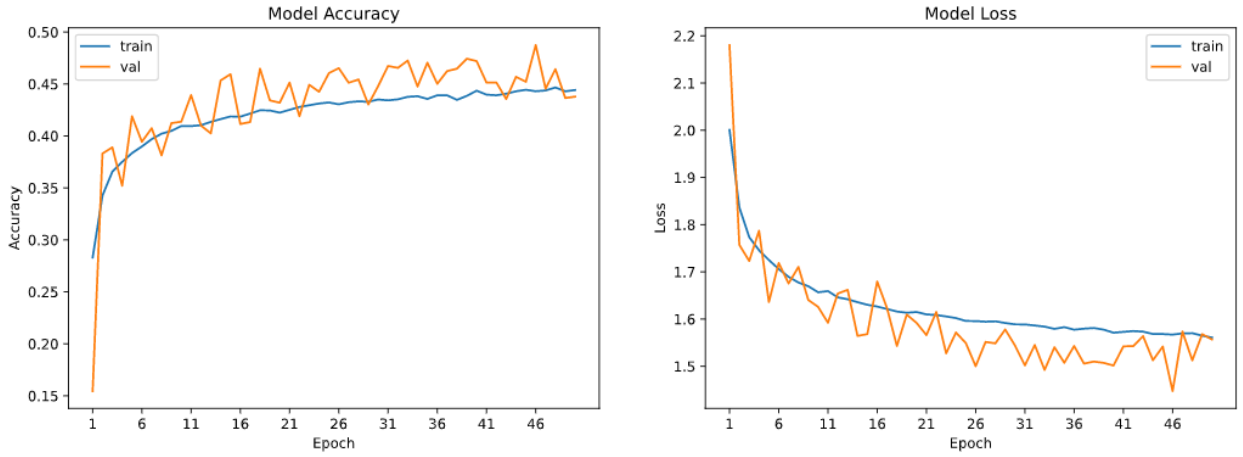

Figura 13: Experimento 0

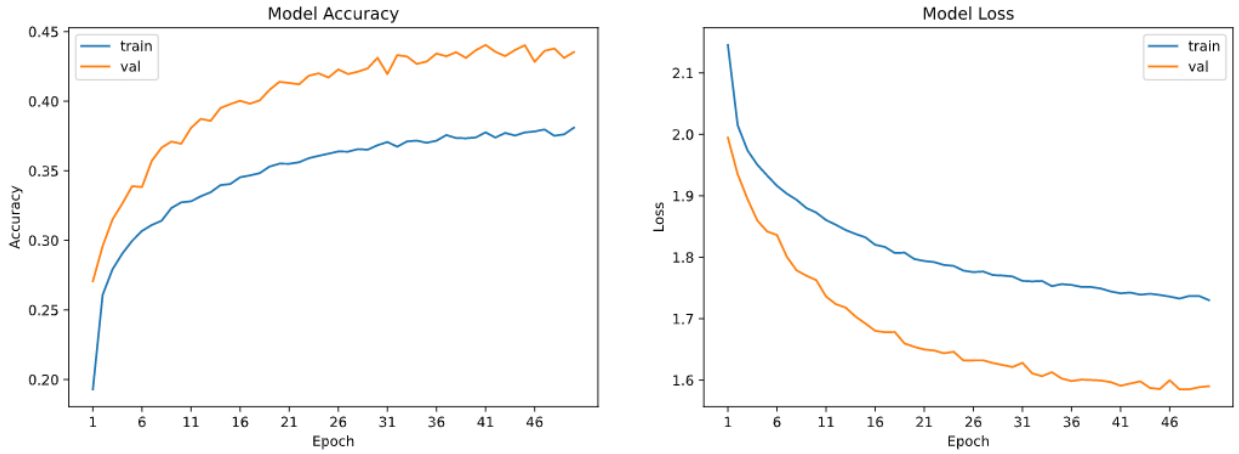

Figura 14: Experimento 1

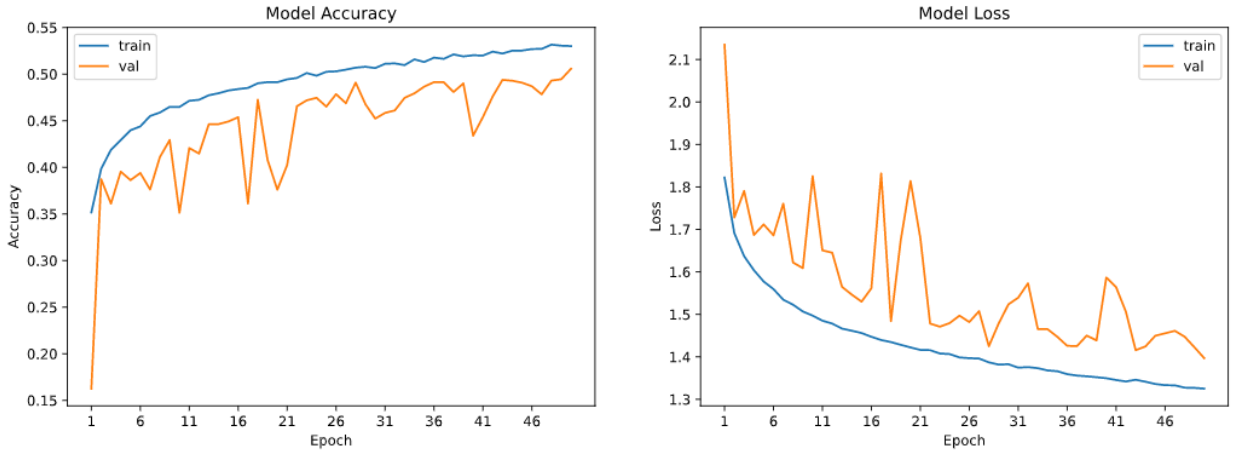

Figura 15: Experimento 2

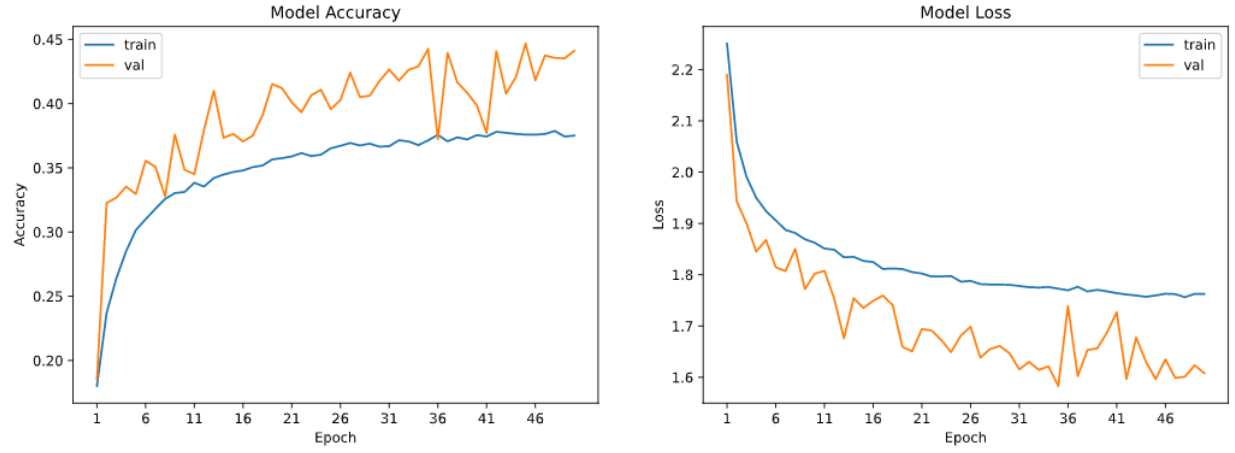

Figura 16: Experimento 3

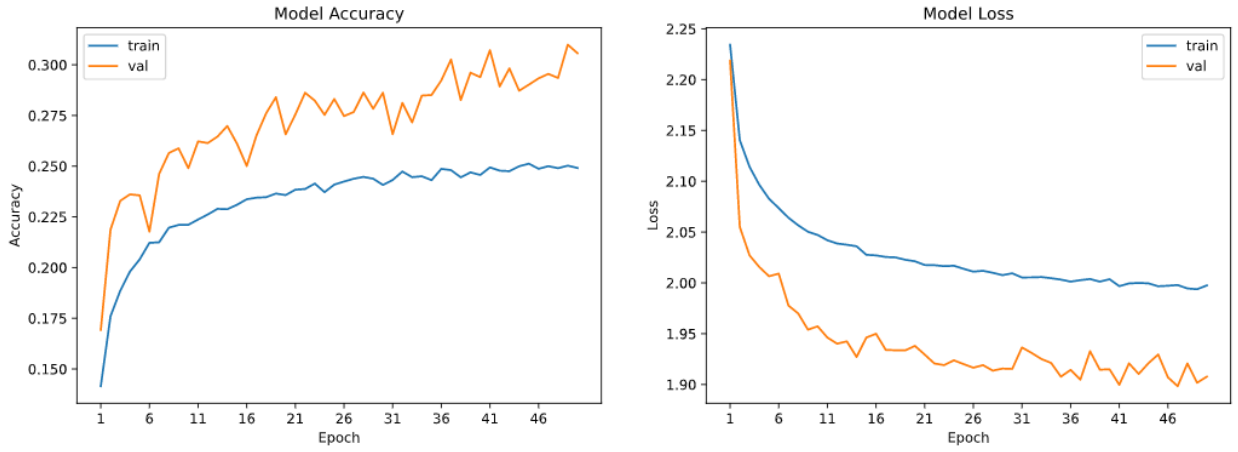

Figura 17: Experimento 4

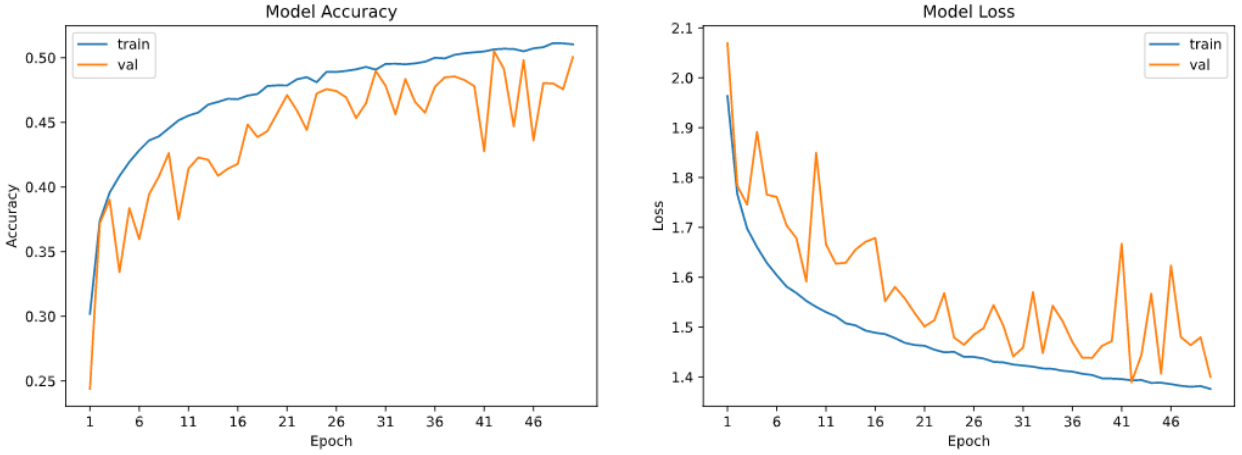

Figura 18: Experimento 5

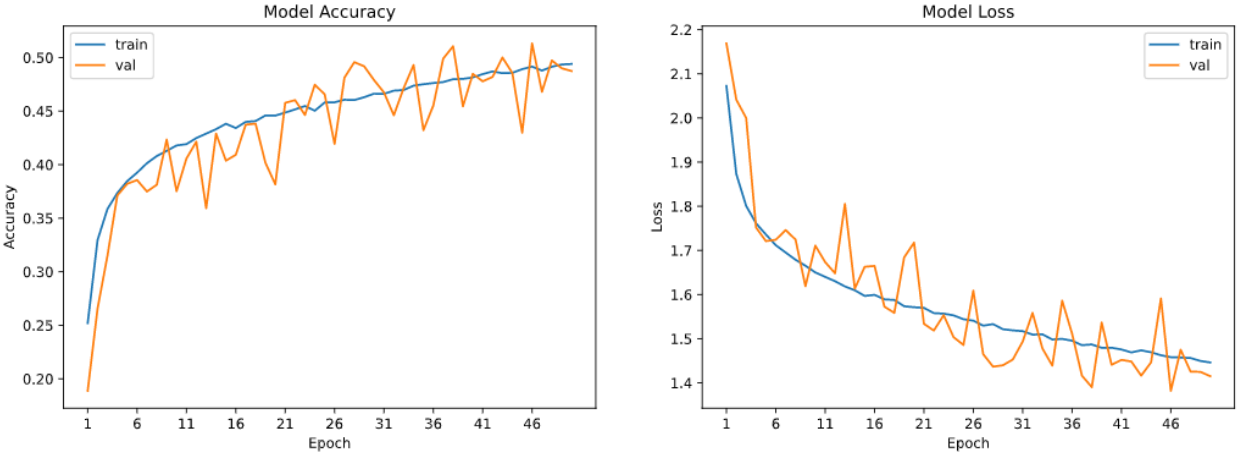

Figura 19: Experimento 6

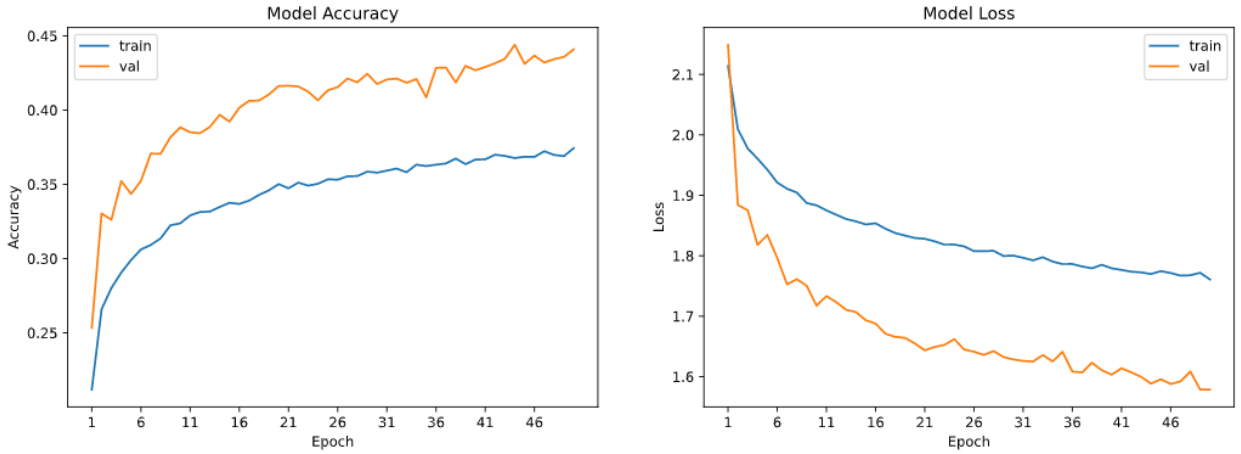

Figura 20: Experimento 7

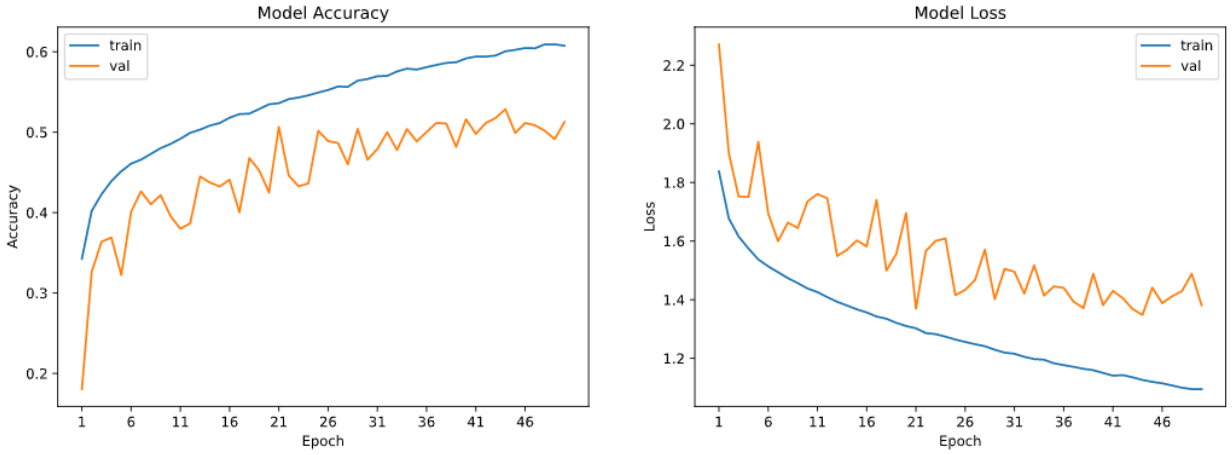

Figura 21: Experimento 8

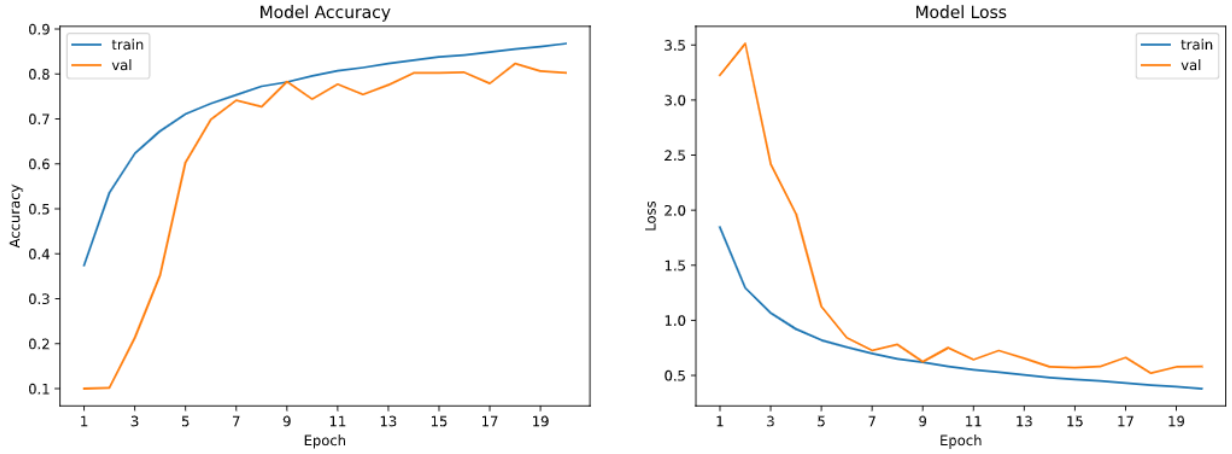

Figura 22: Experimento 9

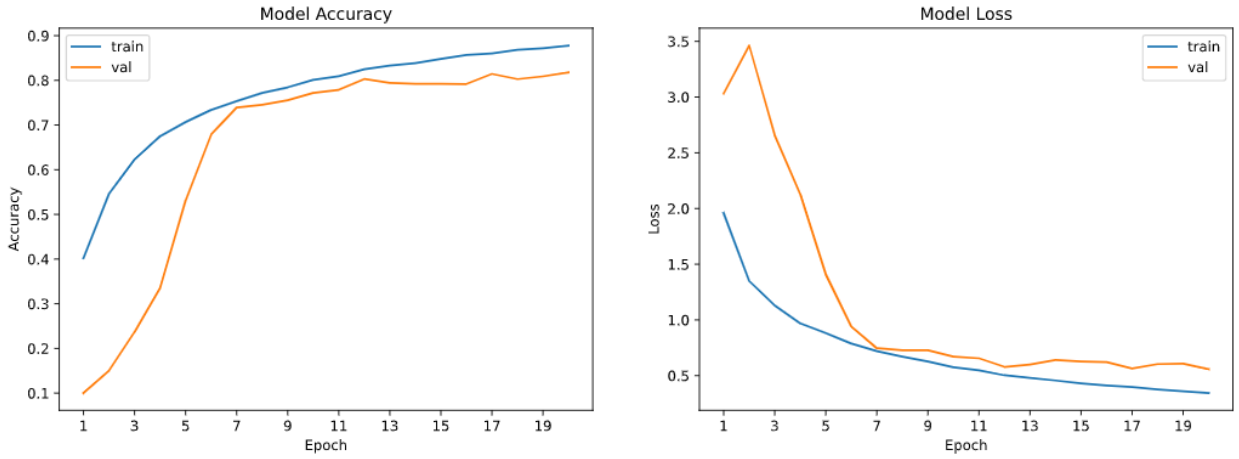

Figura 23: Experimento 10

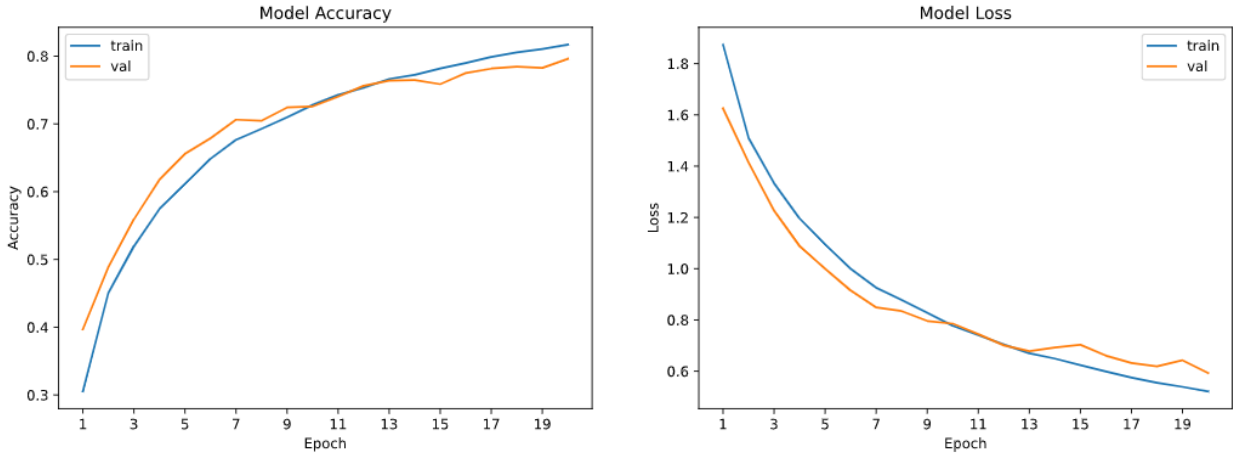

Figura 24: Experimento 11

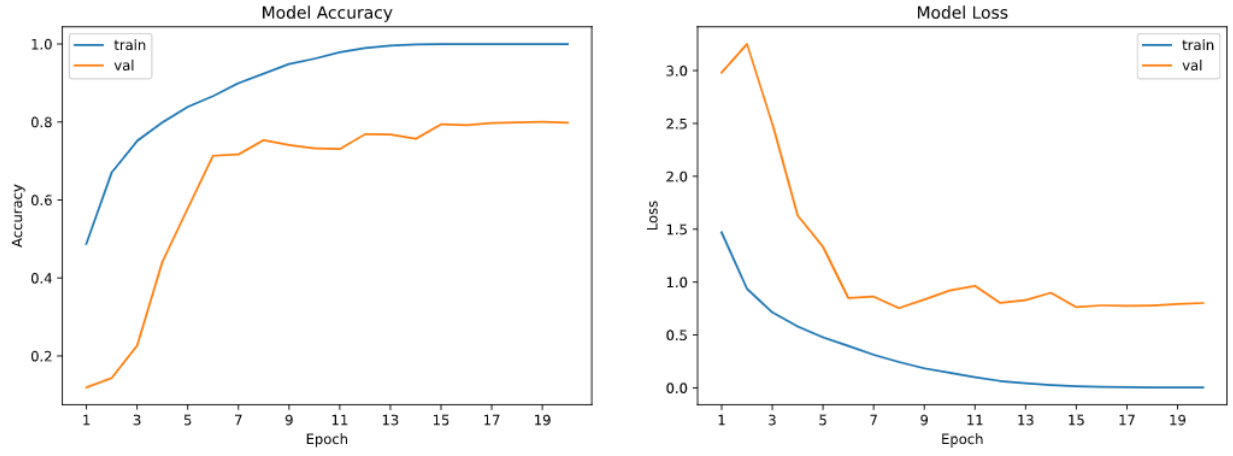

Figura 25: Experimento 12

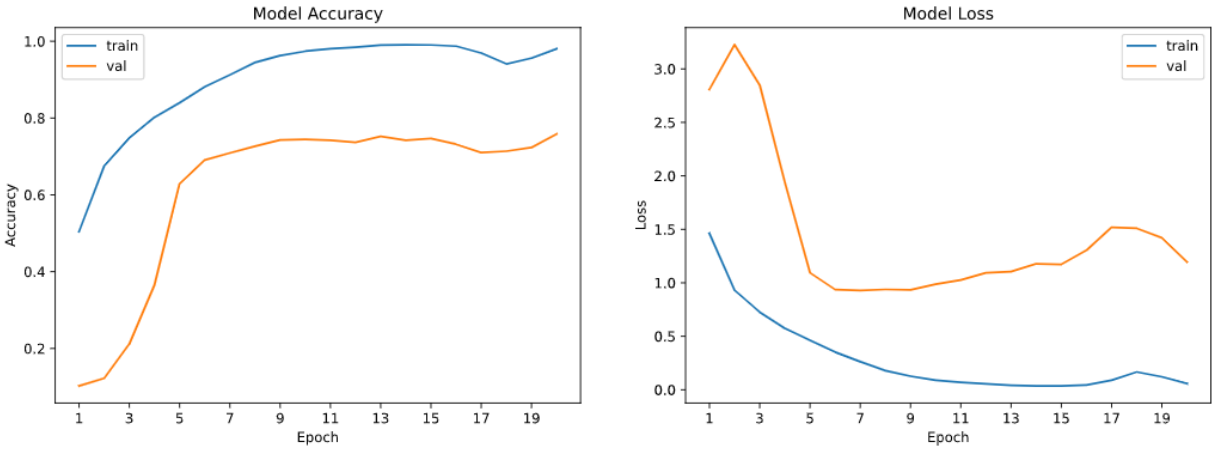

Figura 26: Experimento 13

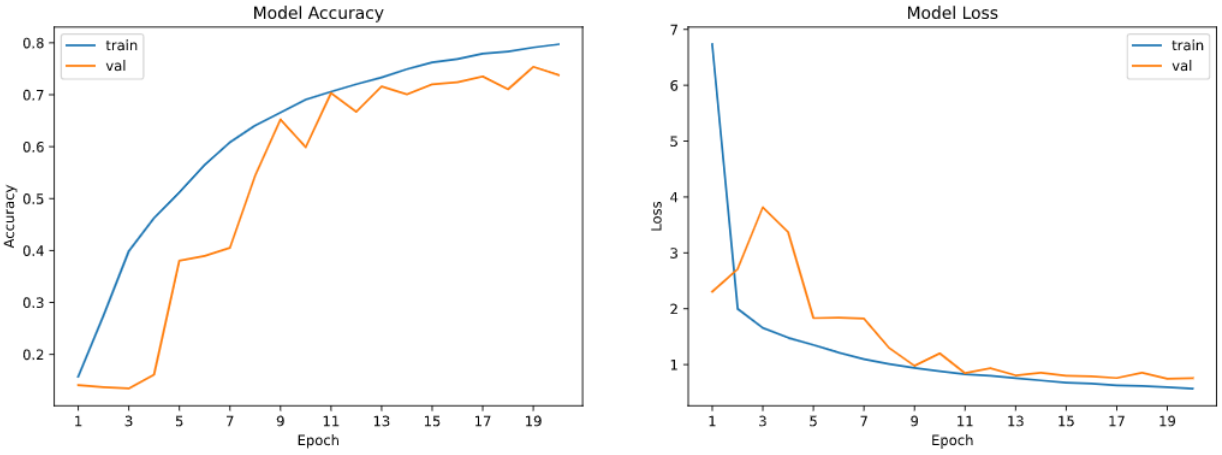

Figura 27: Experimento 14

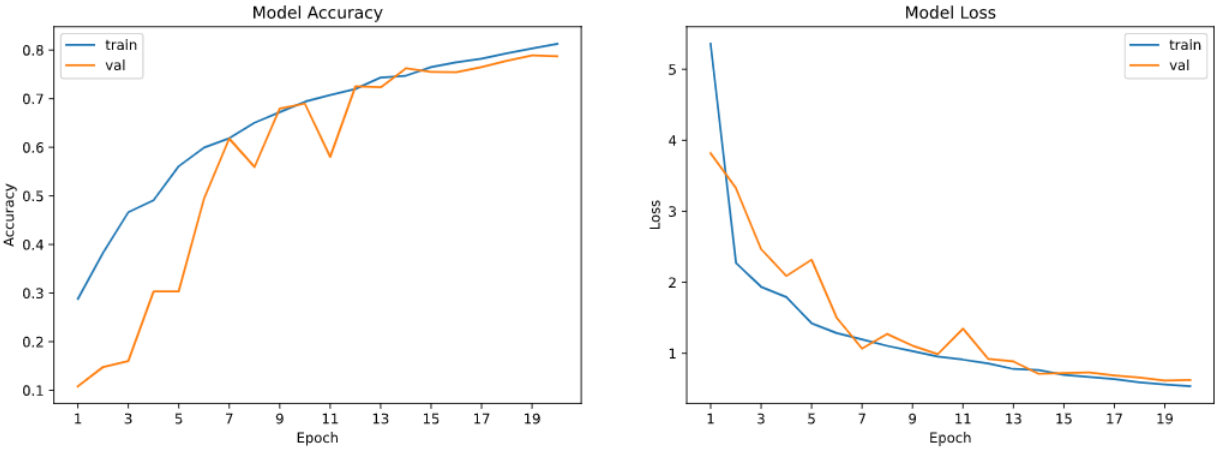

Figura 28: Experimento 15

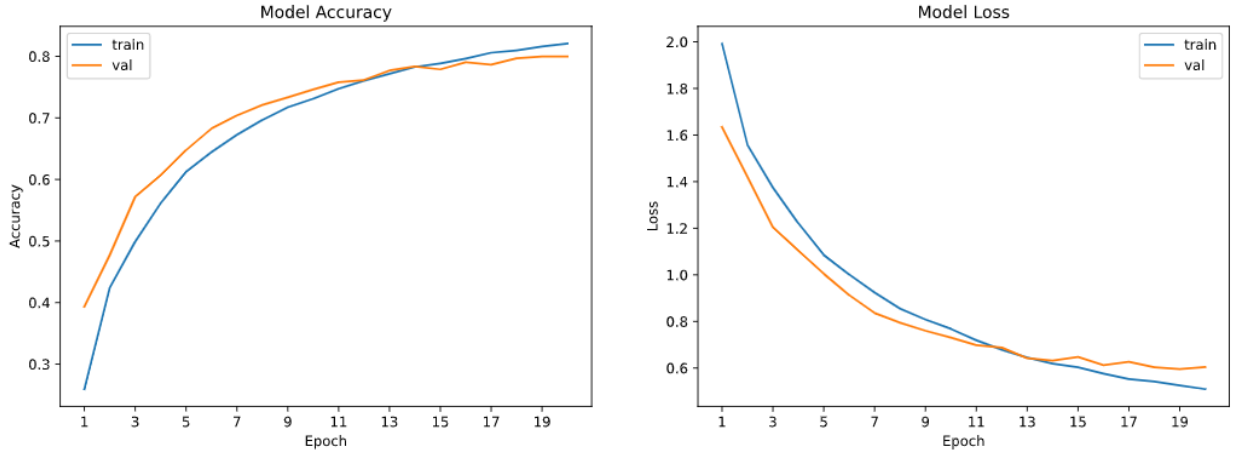

Figura 29: Experimento 16

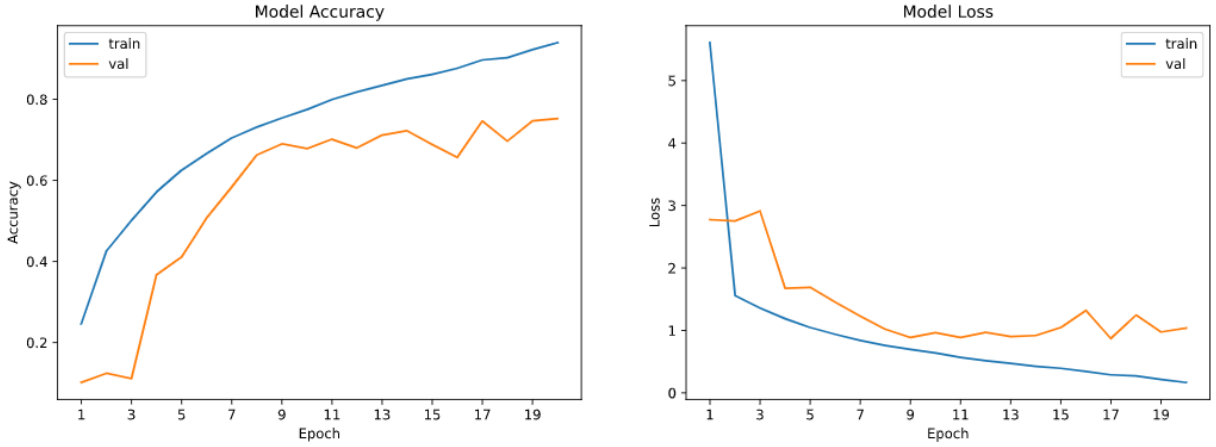

Figura 30: Experimento 17

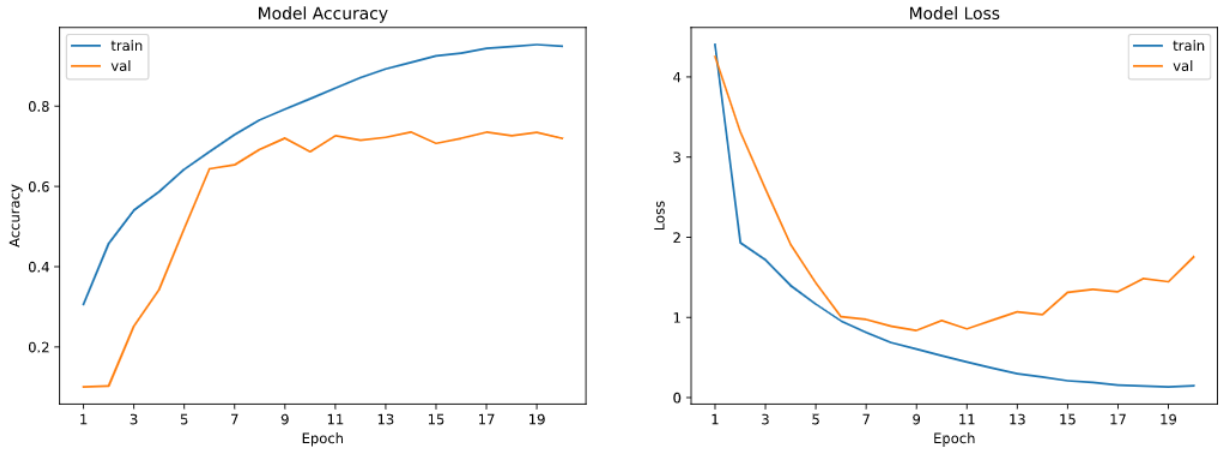

Figura 31: Experimento 18

D. Gráficas de entrenamiento: Traffic Sign

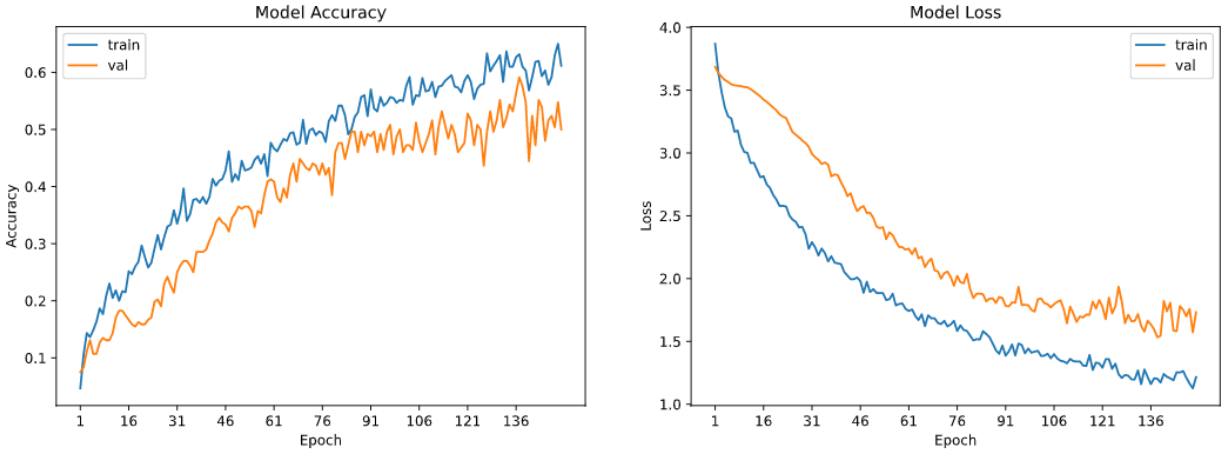

Figura 32: Experimento 19

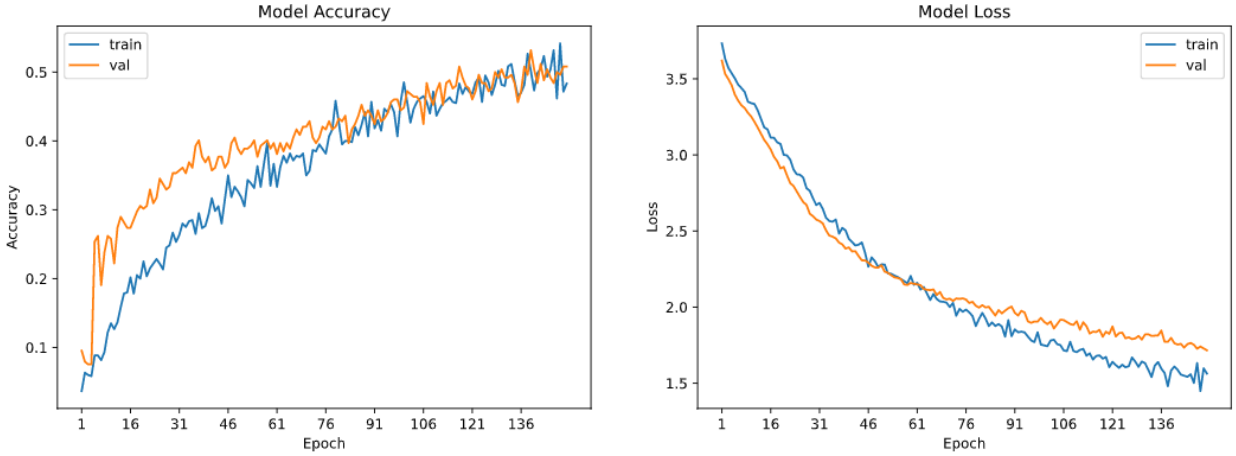

Figura 33: Experimento 20

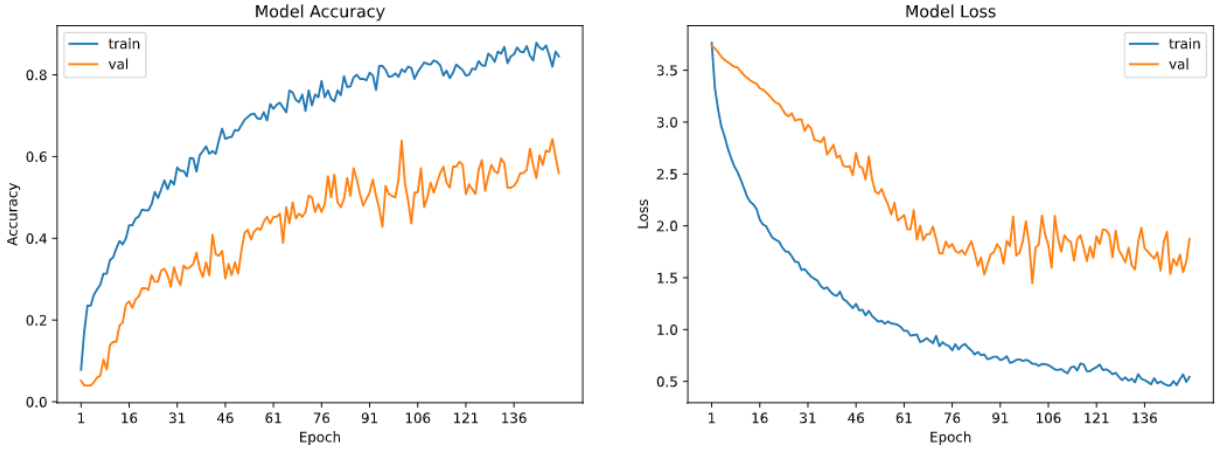

Figura 34: Experimento 21

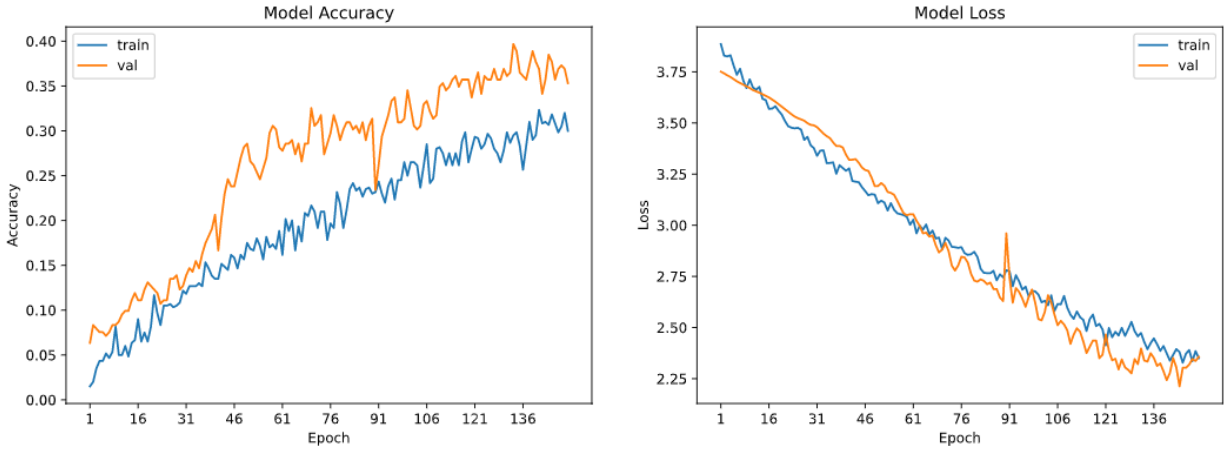

Figura 35: Experimento 22

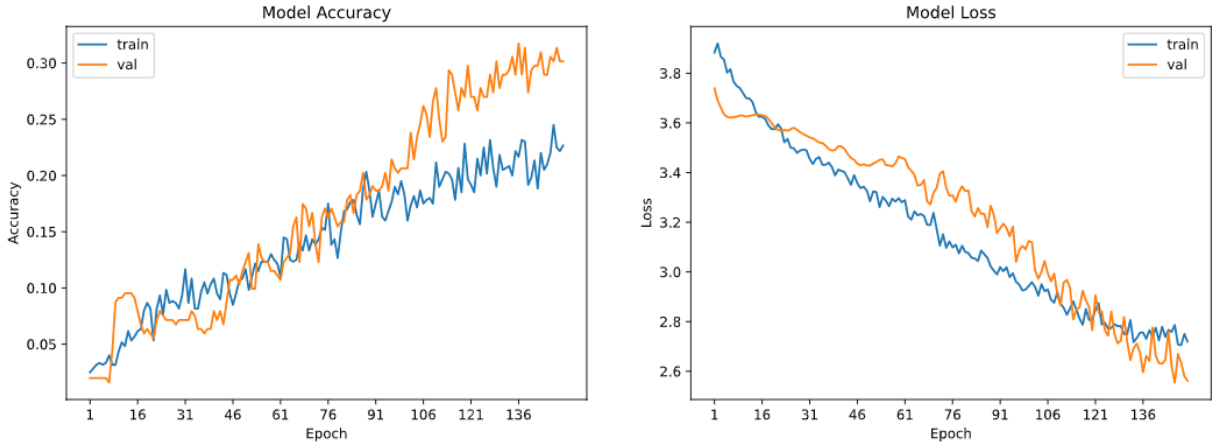

Figura 36: Experimento 23

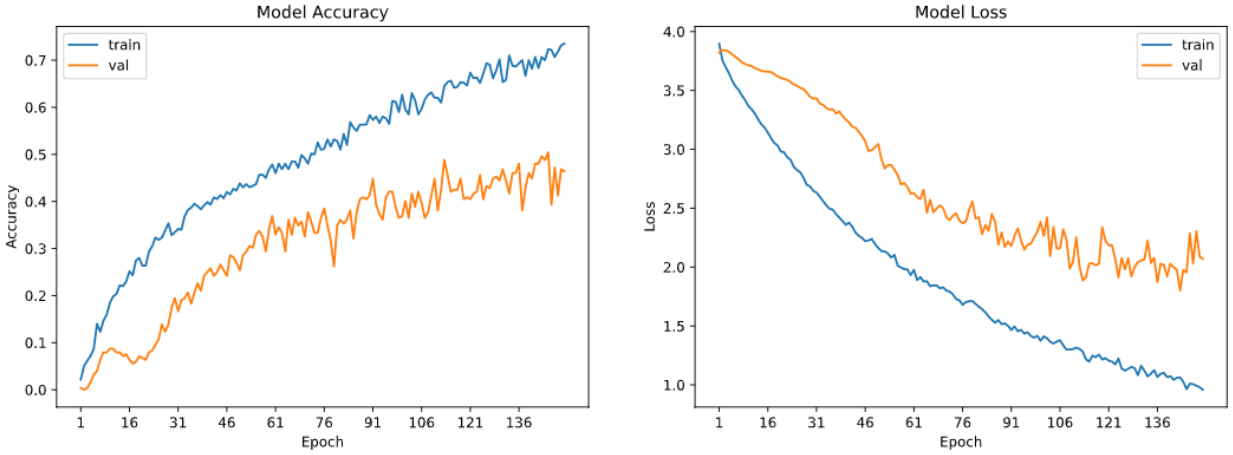

Figura 37: Experimento 24

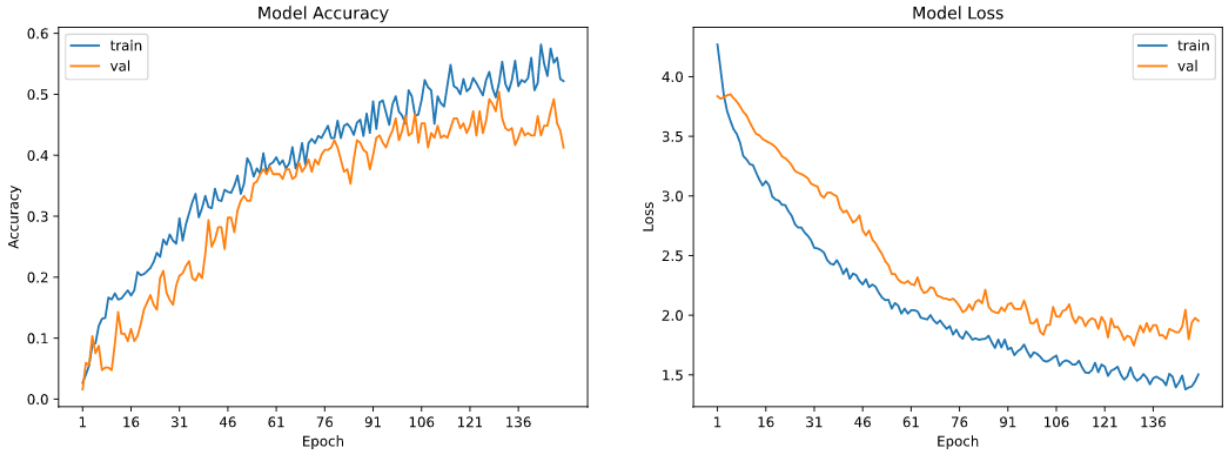

Figura 38: Experimento 25

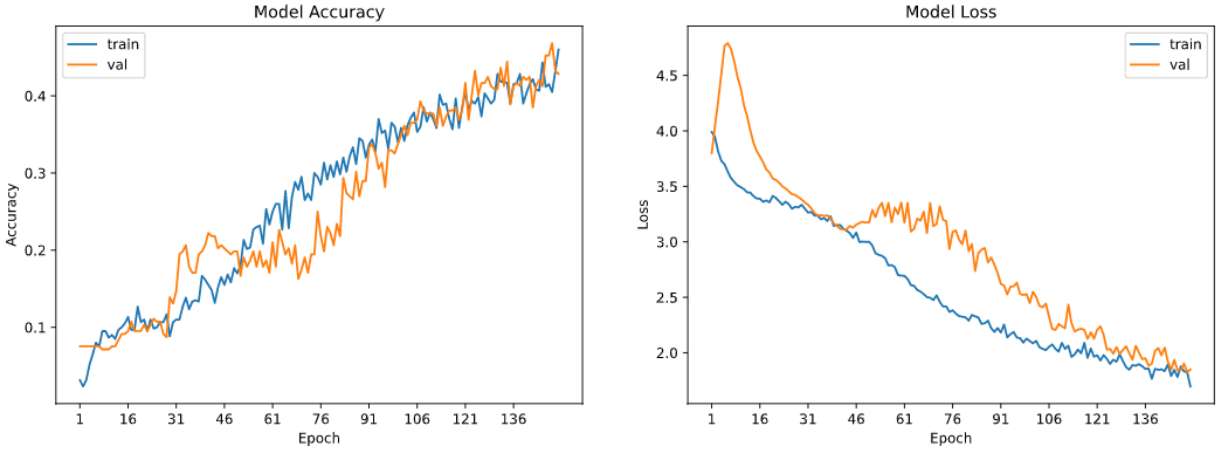

Figura 39: Experimento 26

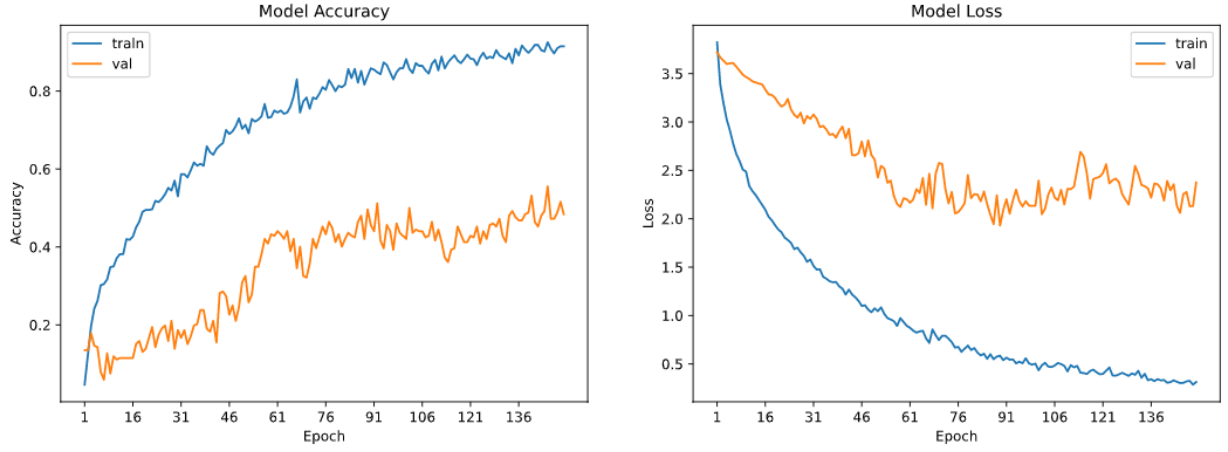

Figura 40: Experimento 27

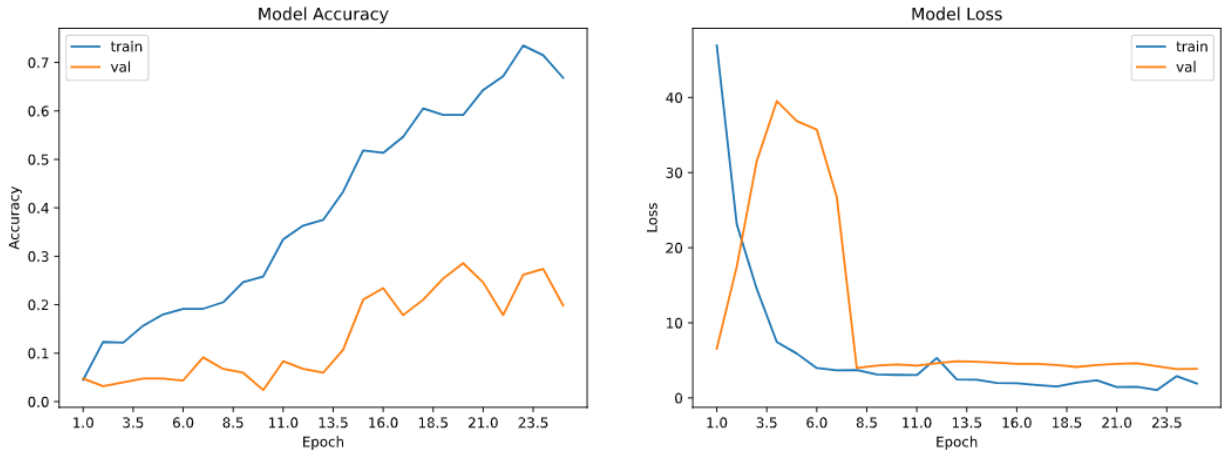

Figura 41: Experimento 28

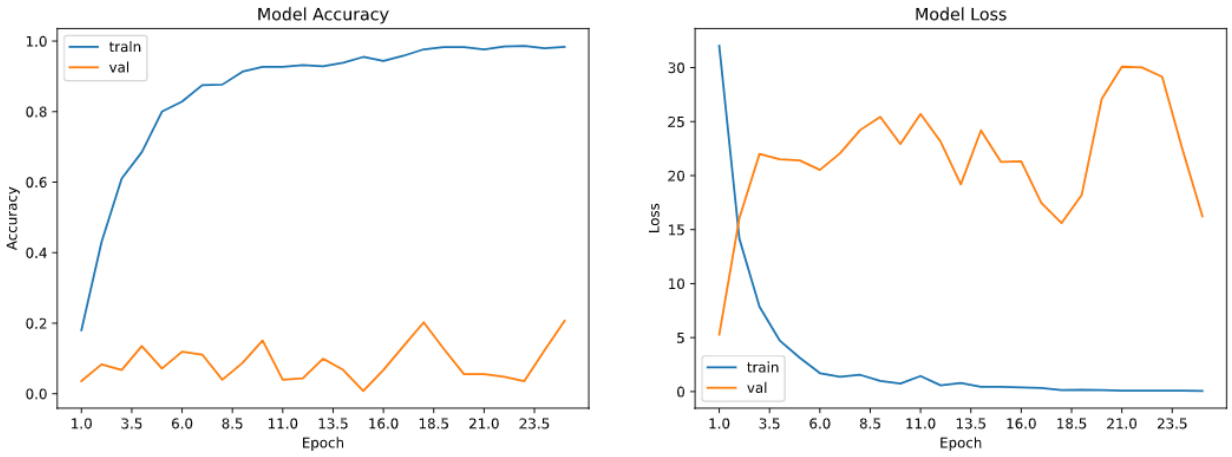

Figura 42: Experimento 29

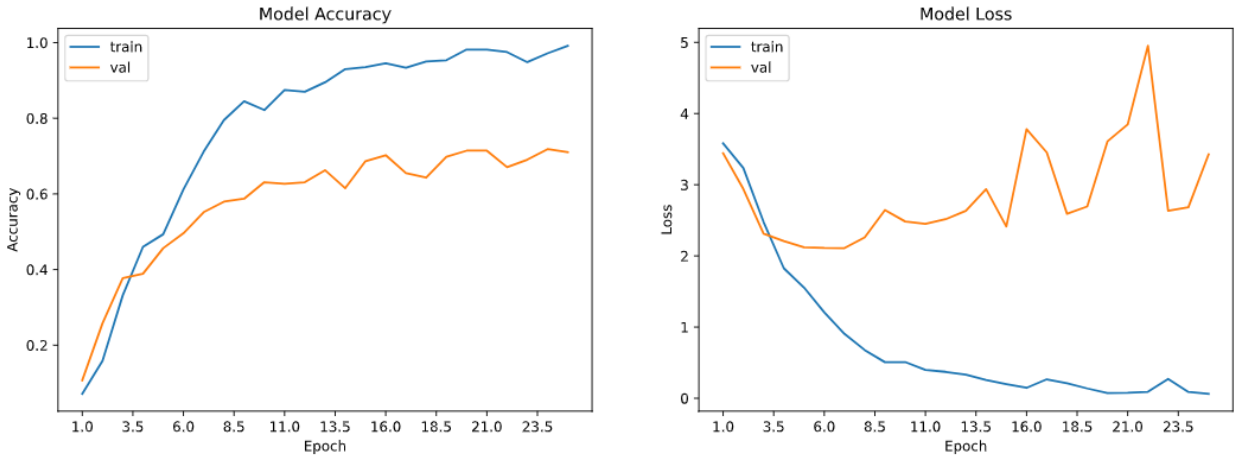

Figura 43: Experimento 30

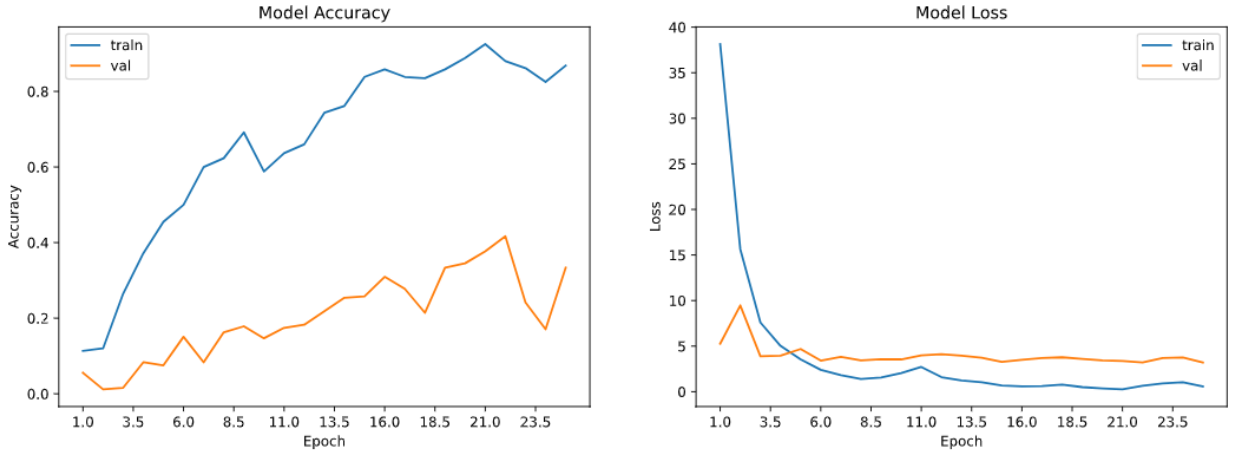

Figura 44: Experimento 31

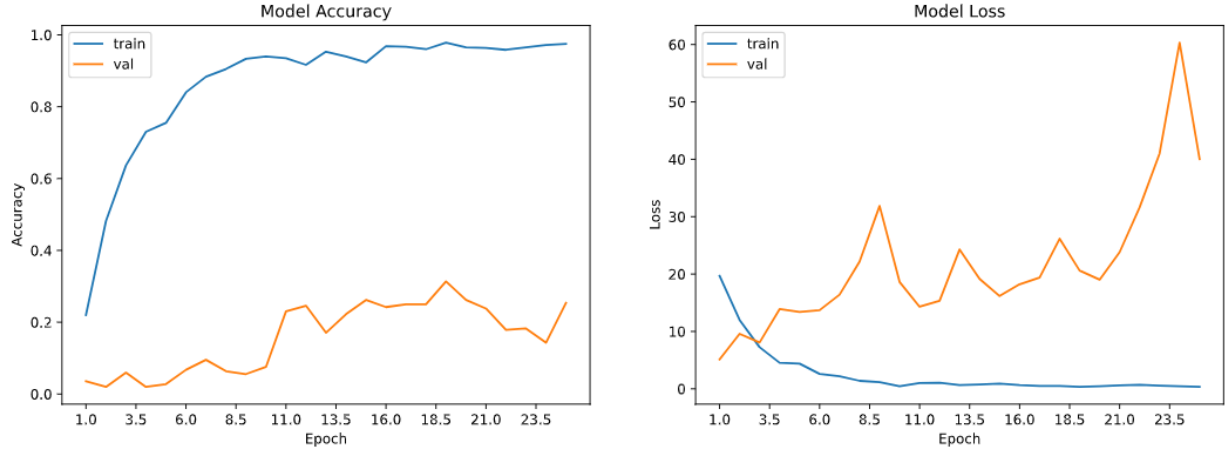

Figura 45: Experimento 32

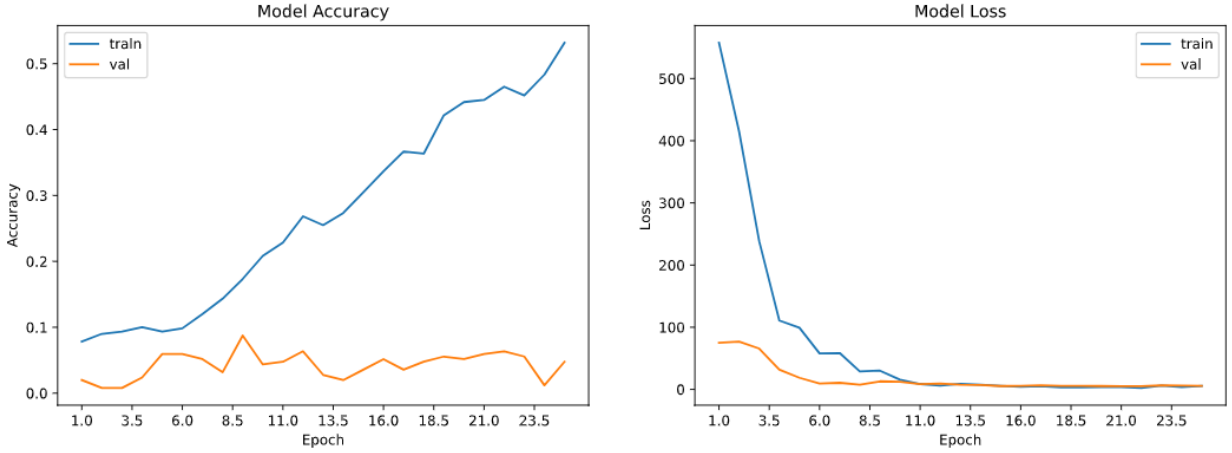

Figura 46: Experimento 34

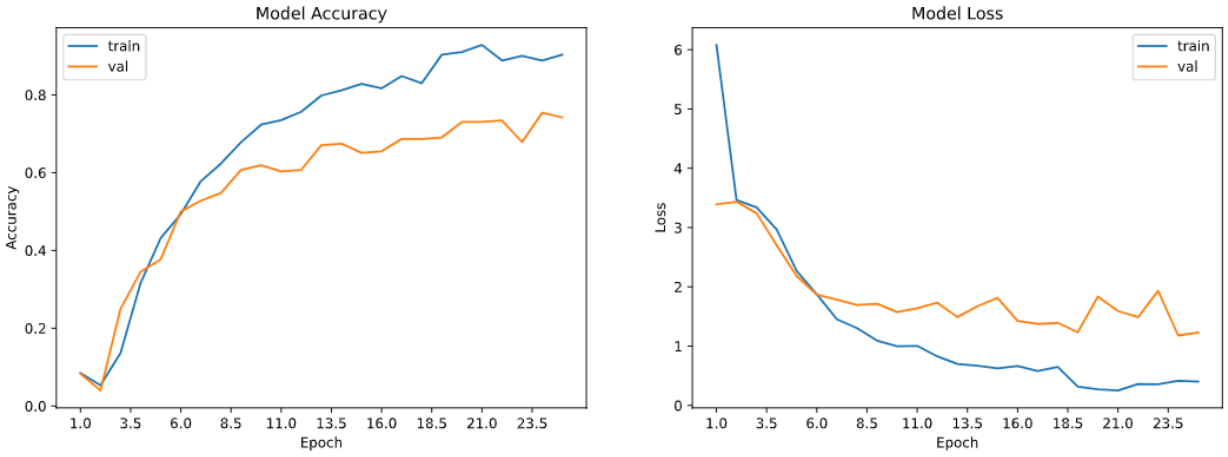

Figura 47: Experimento 35

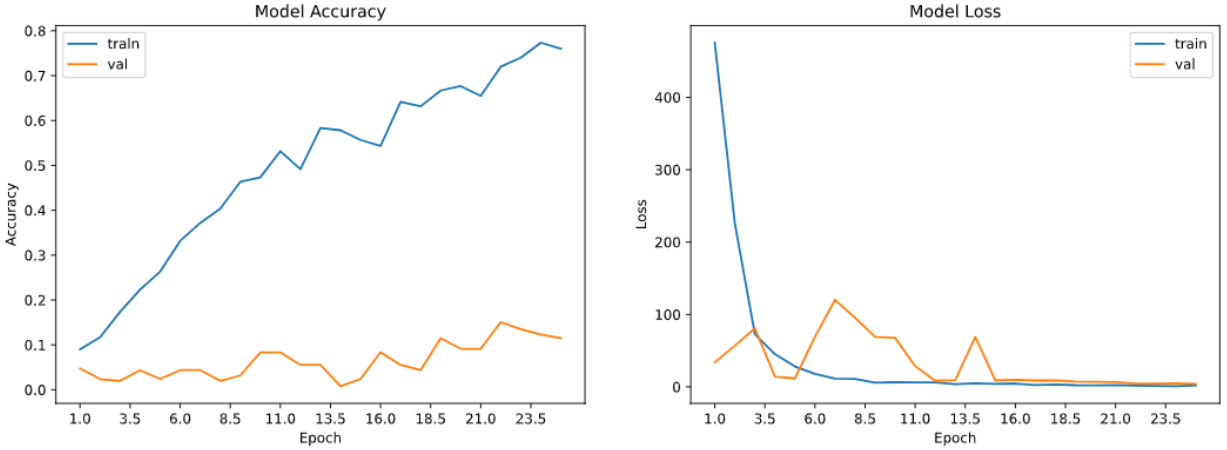

Figura 48: Experimento 36

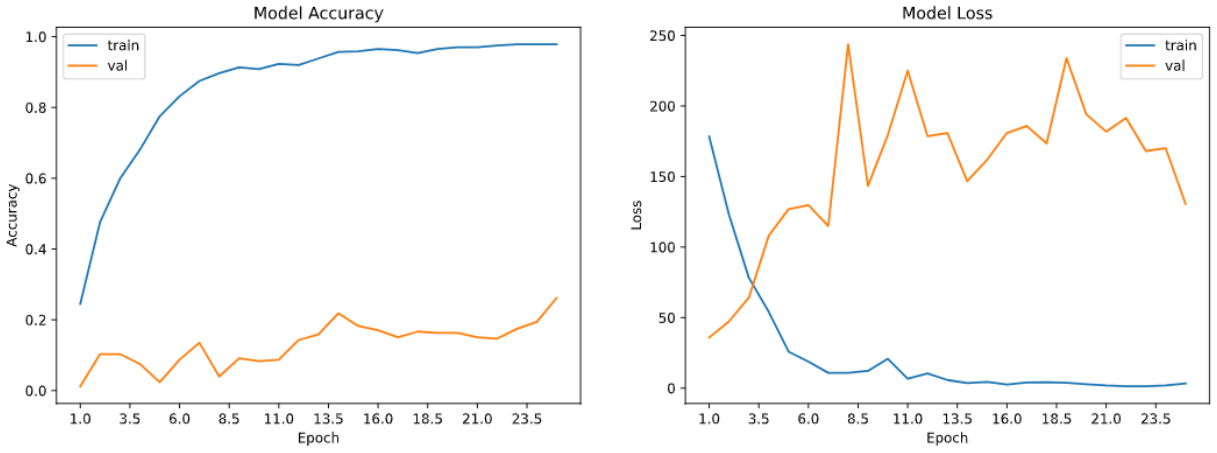

Figura 49: Experimento 37

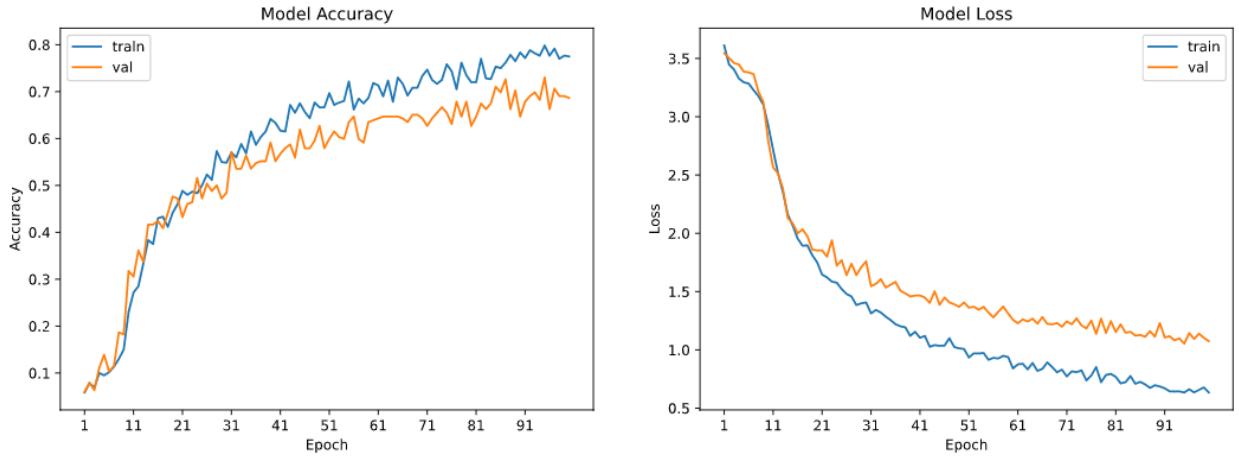

Figura 50: Experimento 38

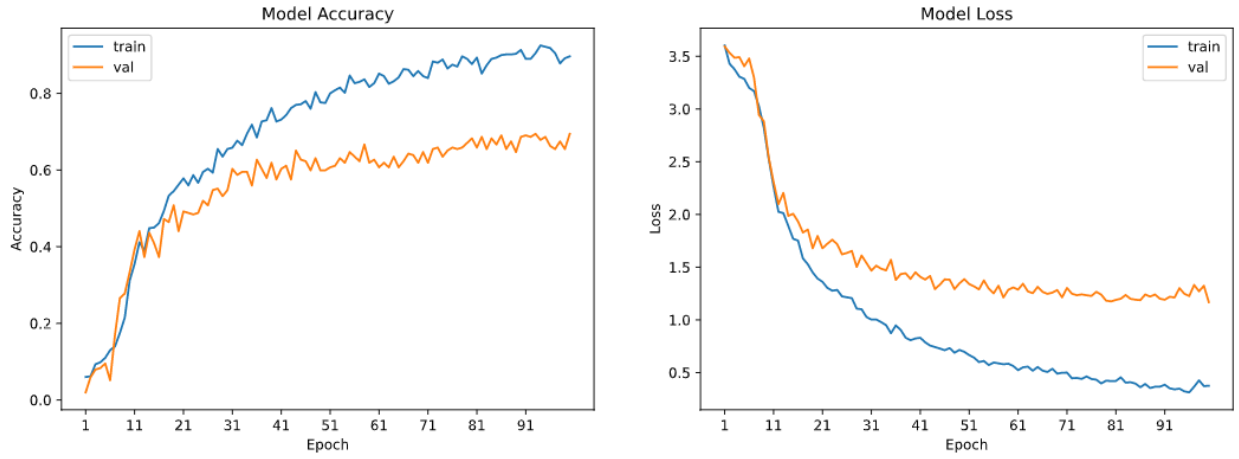

Figura 51: Experimento 39