 # Practico 1

Ejercicio 1

1. Construir un pipeline de clasificación con un modelo Keras MLP. Pueden comenzar con una versión simplicada que sólo tenga una capa de Input donde pasen los valores de las columnas de *one-hot-encodings*.

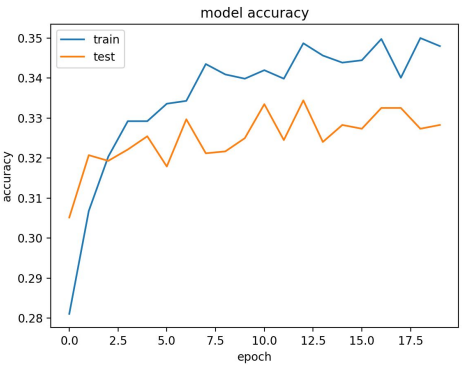
Modelo 1_1

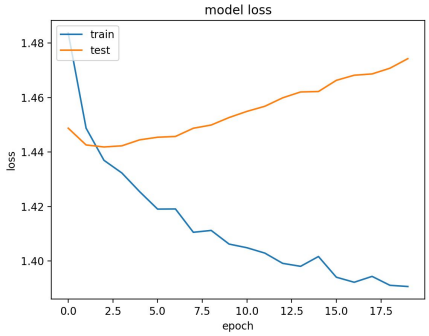
Columnas:
Breed1 (embedding)
Gender, Color1 (one hot encoding)
Age, Fee (numerica normalizadas)

Hiperparametros:
hidden_layer_sizes 64
dropout 0.4
epochs 20
Batch_size 64

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Breed1 (InputLayer)          [(None, 1)]               0
_________________________________________________________________
embedding (Embedding)        (None, 1, 77)             23716
_________________________________________________________________
tf_op_layer_Squeeze (TensorFlow [(None, 77)]           0
_________________________________________________________________
direct_features (InputLayer) [(None, 12)]              0
_________________________________________________________________
concatenate (Concatenate)    (None, 89)                0
_________________________________________________________________
dense (Dense)                (None, 64)                5760
_________________________________________________________________
dropout (Dropout)            (None, 64)                0
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 325
=================================================================
Total params: 29,801
Trainable params: 29,801
Non-trainable params: 0
_________________________________________________________________
```

Evolucion del loss y accuracy entre train y dev.





NOTA: donde dice “test”, debe decir “dev”

Este modelo está overfiteando, el accuracy de dev se plancha y el loss de dev crece luego de la segunda epoch. Hay alguna relación entre el tamaños del train dataset (8465 muestras) con la cantidad de parámetro a entrenar (29801 parámetros)?

2. Entrenar uno o varios modelos (con dos o tres es suficiente, veremos más de esto en el práctico 2). Evaluar los modelos en el conjunto de dev y test.

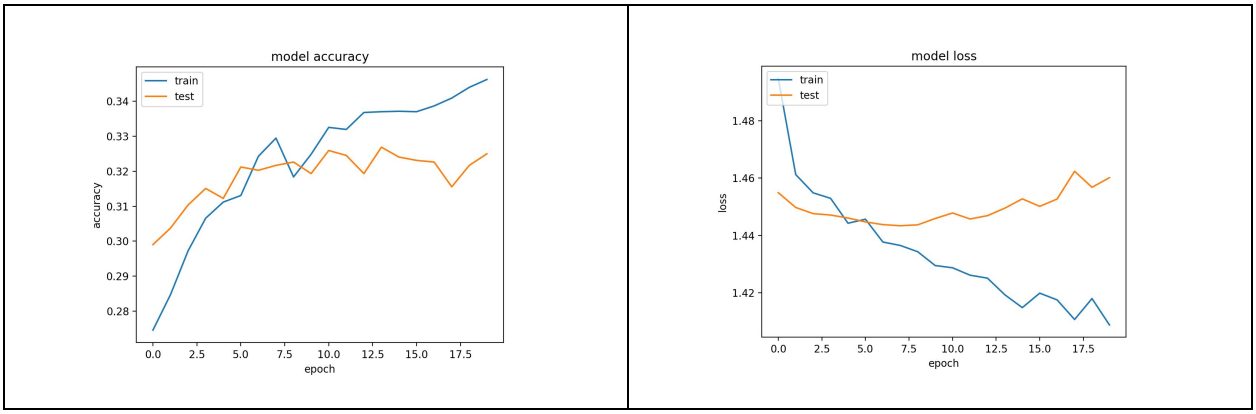
Modelo 1_2_a

Columnas:
Breed1 (embedding)
Gender, Color1 (one hot encoding)
Age, Fee (numerica normalizadas)

Hiperparametros:
hidden_layer_sizes 1024, 512
dropout 0.4, 0.4
epochs 20
Batch_size 64

```
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
Breed1 (InputLayer)             [(None, 1)]          0
__________________________________________________________________________________________________
embedding (Embedding)           (None, 1, 77)        23716       Breed1[0][0]
__________________________________________________________________________________________________
tf_op_layer_Squeeze (TensorFlow [(None, 77)]         0           embedding[0][0]
__________________________________________________________________________________________________
direct_features (InputLayer)    [(None, 12)]         0
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 89)           0           tf_op_layer_Squeeze[0][0]
                                                                 direct_features[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 1024)         92160       concatenate[0][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 1024)         0           dense[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 512)          524800      dropout[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 512)          0           dense_1[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 5)            2565        dropout_1[0][0]
==================================================================================================
Total params: 643,241
Trainable params: 643,241
Non-trainable params: 0
__________________________________________________________________________________________________
```

Evolucion del loss y accuracy entre train y dev.



NOTA: donde dice “test”, debe decir “dev”

Al igual que el modelo modelo_1_1, este modelo está overfiteando aun agregando más capas aunque en un epoch mayor, el accuracy de dev se plancha y el loss de dev crece luego de la quinta epoch. Nos hacemos la misma pregunta del modelo anterior.

Modelo 1_2_b

Columnas:
Breed1 (embedding)
Gender, Color1 (one hot encoding)

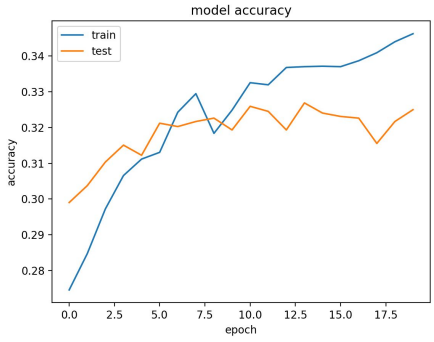
Age, Fee (numerica normalizadas)

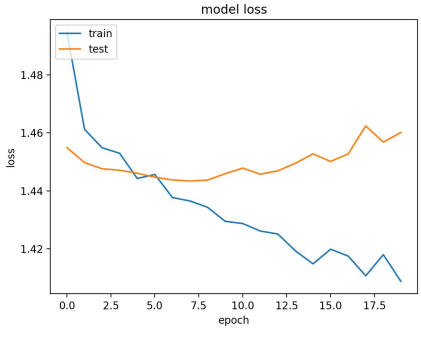
Hiperparametros:
hidden_layer_sizes 128, 64
dropout 0.4, 0.4
epochs 20
Batch_size 64

Layer (type)	Output Shape	Param #	Connected to
Breed1 (InputLayer)	[(None, 1)]	0	
embedding (Embedding)	(None, 1, 77)	23716	Breed1[0][0]
tf_op_layer_Squeeze (TensorFlow	[(None, 77)]	0	embedding[0][0]
direct_features (InputLayer)	[(None, 12)]	0	
concatenate (Concatenate)	(None, 89)	0	tf_op_layer_Squeeze[0][0] direct_features[0][0]
dense (Dense)	(None, 128)	11520	concatenate[0][0]
dropout (Dropout)	(None, 128)	0	dense[0][0]
dense_1 (Dense)	(None, 64)	8256	dropout[0][0]
dropout_1 (Dropout)	(None, 64)	0	dense_1[0][0]
dense_2 (Dense)	(None, 5)	325	dropout_1[0][0]

Total params: 43,817
Trainable params: 43,817
Non-trainable params: 0

Evolucion del loss y accuracy entre train y dev.





NOTA: donde dice “test”, debe decir “dev”

Al igual que el modelo modelo_1_1 y modelo_1_2_a, este modelo está overfiteando aun volviendo a bajar la cantidad de parámetros, el epoch donde comienza el loss de dev a subir es similar al del modelo_1_2_a, el accuracy de dev también se plancha.

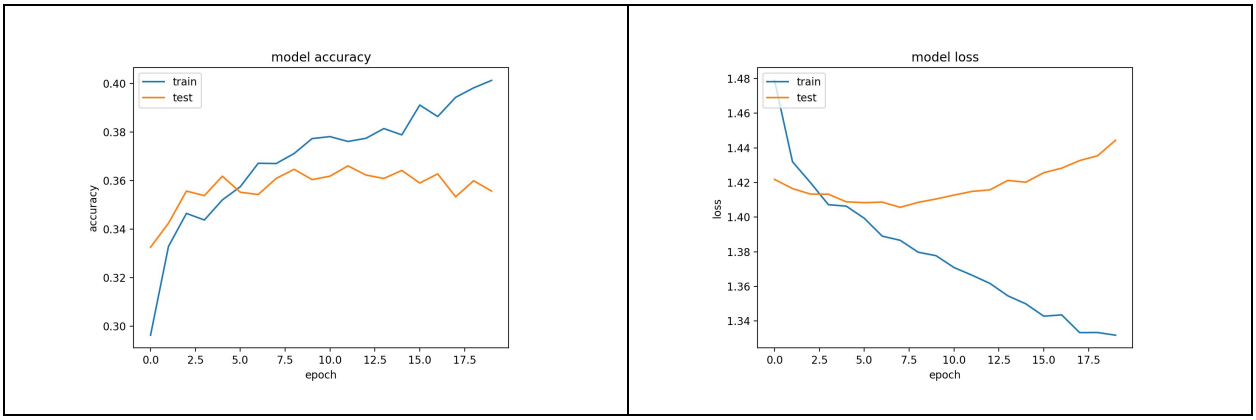
Ejercicio 2

1. Utilizar el mismo modelo anterior y explorar cómo cambian los resultados a medida que agregamos o quitamos columnas.

Modelo_2_1_a

Al modelo modelo_1_2_b anterior se le agregaron las columnas one-hot-encoding “MaturitySize FurLength Vaccinated Dewormed Sterilized Health”

Evolucion del loss y accuracy entre train y dev.



NOTA: donde dice “test”, debe decir “dev”

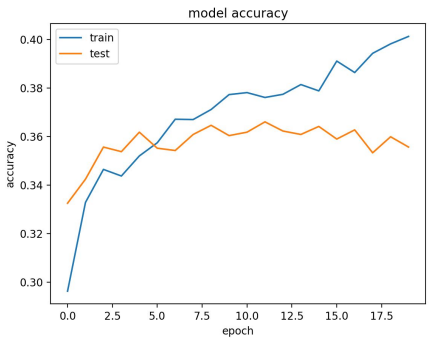
Mejoró el accuracy, aunque seguimos con el mismo problema que se plancha y el loss sube en el dev dataset.

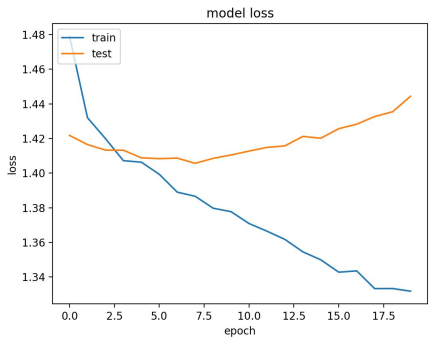
Modelo_2_1_b

Al modelo modelo_1_2_b anterior se le agregaron la columna embedding “Breed2”.

```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
Breed1 (InputLayer)             [(None, 1)]          0
__________________________________________________________________________________________________
Breed2 (InputLayer)             [(None, 1)]          0
__________________________________________________________________________________________________
embedding (Embedding)           (None, 1, 77)        23716       Breed1[0][0]
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, 1, 77)        23716       Breed2[0][0]
__________________________________________________________________________________________________
tf_op_layer_Squeeze (TensorFlow [(None, 77)]         0           embedding[0][0]
__________________________________________________________________________________________________
tf_op_layer_Squeeze_1 (TensorFl [(None, 77)]         0           embedding_1[0][0]
__________________________________________________________________________________________________
direct_features (InputLayer)    [(None, 31)]         0
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 185)          0           tf_op_layer_Squeeze[0][0]
                                                                 tf_op_layer_Squeeze_1[0][0]
                                                                 direct_features[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 128)          23808       concatenate[0][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 128)          0           dense[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 64)           8256        dropout[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 64)           0           dense_1[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 5)            325         dropout_1[0][0]
==================================================================================================
Total params: 79,821
Trainable params: 79,821
Non-trainable params: 0
__________________________________________________________________________________________________
```

Evolucion del loss y accuracy entre train y dev.





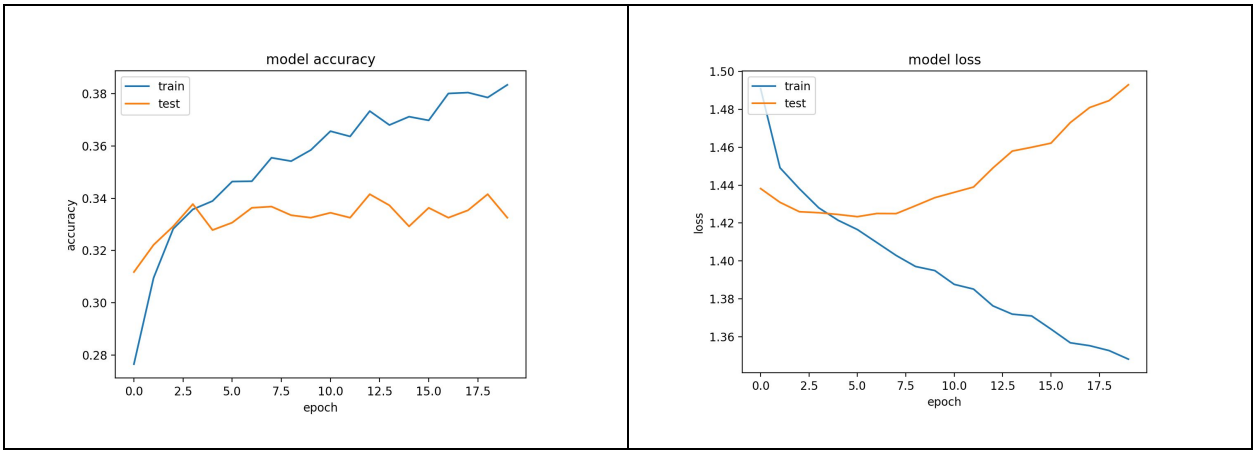
NOTA: donde dice “test”, debe decir “dev”

El accuracy no mejoró significativamente comparado con el modelo_2_1_a.

Modelo_2_1_c

Al modelo modelo_2_1_a se le saco la columna one-hot-encoding “Sterilized”. Hipótesis es que la esterilización no incide tanto como la vacunacion, desparasitacion y salud genera para tomar la decisión de adopción.

Evolucion del loss y accuracy entre train y dev.



NOTA: donde dice “test”, debe decir “dev”

En relación al modelo_2_1_a el accuracy empeoro, con estos pobres resultados vamos a decir que nuestra hipótesis era falsa.

2. Volver a ejecutar una exploración de hyperparámetros teniendo en cuenta la información que agregan las nuevas columnas.

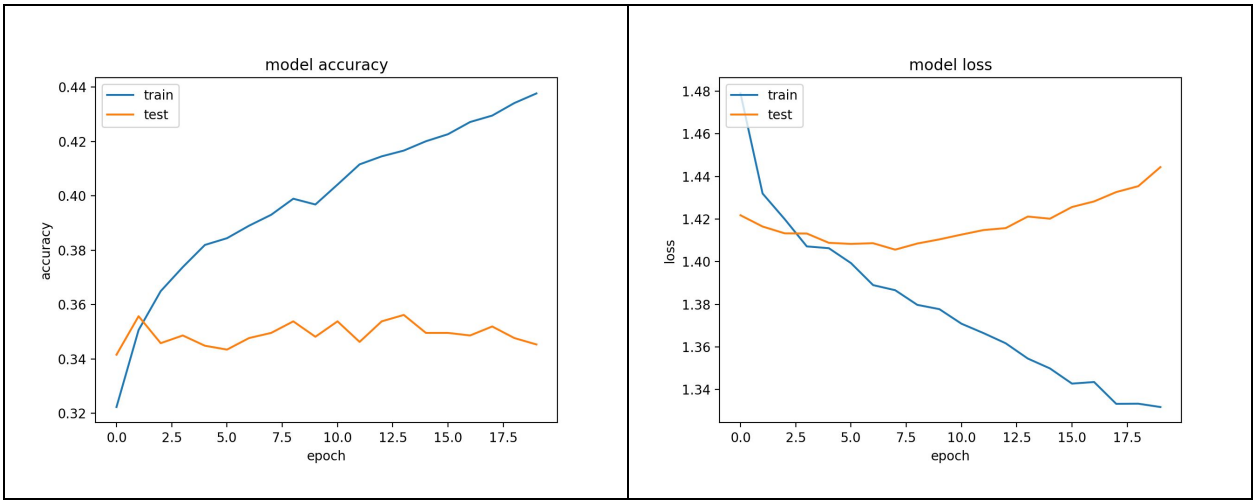
Modelo_2_2_a

Se usa como base el modelo_2_1_a, cambiamos a cada dropout de 0.4 a 0.2. Esperamos peor accuracy.

Columnas:
Breed1 (embedding)
Gender, Color1, MaturitySize FurLength Vaccinated Dewormed Sterilized Health (one hot encoding)
Age, Fee (numerica normalizadas)

Hiperparametros:
hidden_layer_sizes 128, 64
dropout 0.2, 0.2

epochs 20
Batch_size 64
Evolucion del loss y accuracy entre train y dev.



NOTA: donde dice “test”, debe decir “dev”

En efecto el accuracy se aleja de 0.36, en cambio en el modelo modelo_2_1_a el accuracy se acerca a 0.36.

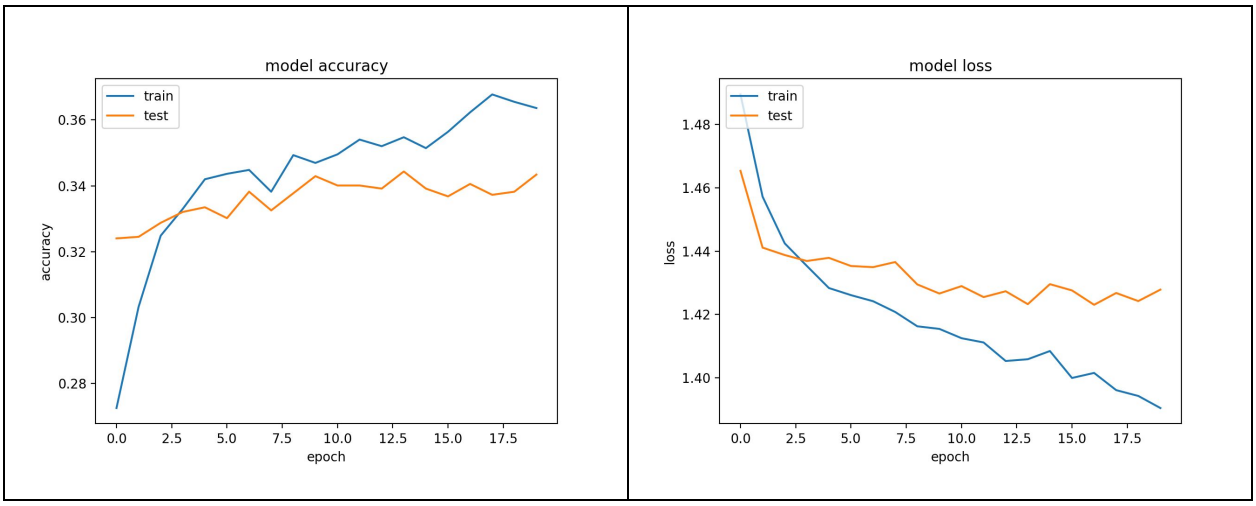
Modelo_2_2_b

Se usa como base el modelo_2_1_a, agregamos una capa de 256 antes de las capas 128, 64 y agrgamos capas de 32, 64, 128 y 256 luego de las capas 128, 64, mismos dropouts.
Esperamos por lo menos un accuracy similar.

Columnas:
Breed1 (embedding)
Gender, Color1, MaturitySize FurLength Vaccinated Dewormed Sterilized Health (one hot encoding)
Age, Fee (numerica normalizadas)

Hiperparametros:
hidden_layer_sizes 256, 128, 64, 32, 64, 128, 256
dropout 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4
epochs 20
Batch_size 64

Evolucion del loss y accuracy entre train y dev.



NOTA: donde dice “test”, debe decir “dev”

La curva de dev acompaña mejor a la de train en esta arquitectura en relación a todas las otras curvas que hemos visto, sin embargo el accuracy sigue siendo más bajo que el modelo modelo_2_1_b (está entre 0.34-0.36) y el loss de dev deja de bajar, se plancha.

Conclusion final practico 1

La búsqueda de hyperparámetros y columnas se realizó manualmente y es tedioso y tantear en el aire, se puede hacer muchos experimentos usando exploración random para la búsqueda de hyperparámetros y columnas.

Las columnas correctas influyeron más que el cambio de arquitectura una vez seleccionadas las columnas. Aunque con los pobres resultados obtenidos en general no puedo confiar en esta conclusión en particular.

No logre entender porque el accuracy se plancha y el loss sube luego de ciertas epochs. Son las pocas muestras en relación a los parámetros a entrenar?

Faltó realizar una exploración random y ejecutar cada modelo varias veces con distintas inicializaciones, luego evaluarlos en test y poder quedas con el de menor variación en su accuracy.

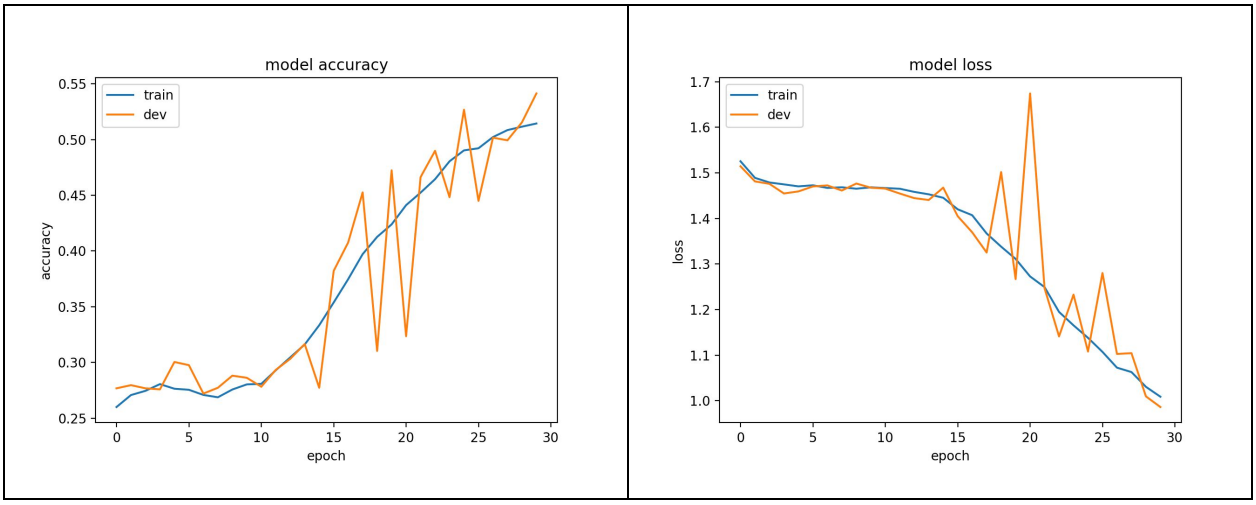
 # Practico 2

Modelo_2_1

Hiperparametros:
filter_widths 2, 3, 5
filter_count 64
hidden_layer_sizes 256, 128, 64, 32, 64, 128, 256
dropout 0.4, 0.4, 0.4, 0.4, 0.4, 0.4, 0.4
epochs 30
batch_size 128
filter_widths 2 3 5
filter_count 64

Columnas:
Breed1 (embedding)
Gender, Color1, MaturitySize FurLength Vaccinated Dewormed Sterilized Health (one hot encoding)
Description (word embedding)

Evolucion del loss y accuracy entre train y dev.



Este modelo se base en el modelo_2_2_b del práctico anterior agregando description como feature, el accuracy aumentó considerablemente. Y las curvas de dev siguen mejor a test.

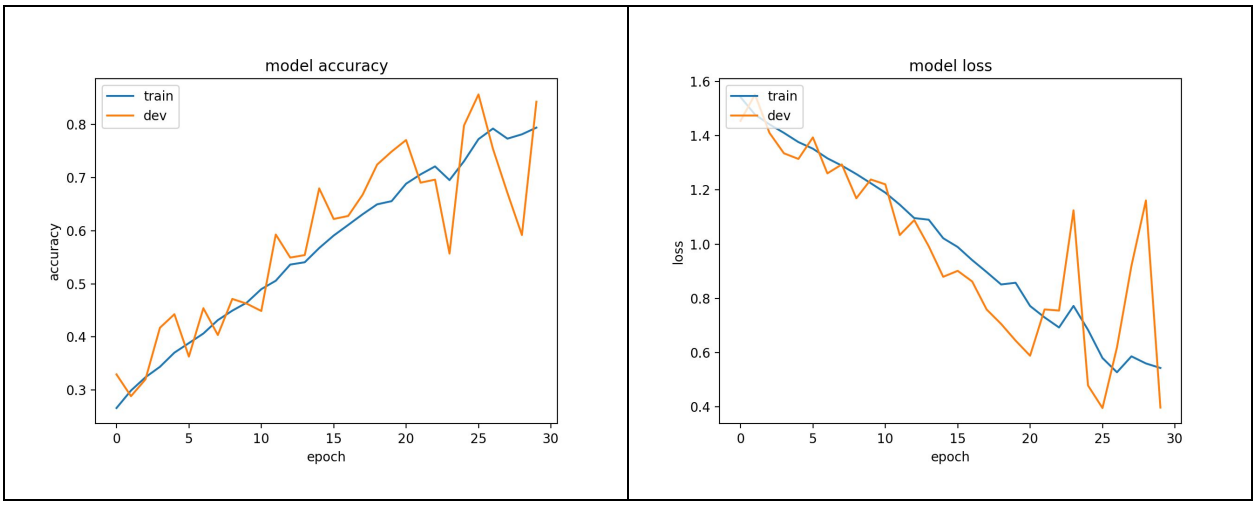
Modelo_2_2

Hiperparametros:
hidden_layer_sizes 128, 64

dropout 0.4, 0.4
epochs 30
batch_size 128
filter_widths 2 3 5
filter_count 64

Columnas:
Breed1 (embedding)
Gender, Color1, MaturitySize FurLength Vaccinated Dewormed Sterilized Health (one hot encoding)
Description (word embedding)

Evolucion del loss y accuracy entre train y dev.



En este modelo el accuracy subió bastante eliminando capas ocultas. Tiene sentido preguntarse el porqué de esos picos al final de las curvas de dev?, sera por el learning rate?

Conclusion final practico 2

Vemos que la columna description junto a word embeddings le dio un boost importante al modelo. Lo interesante del modelo_2_2 e que pudimos achicar las capas ocultas y obtener un salto grande en el accuracy.

Con las capas de convolución los parámetros de entrenamiento subieron a más de un millón, por lo que la pregunta que hacía en el práctico 1 (“Son las pocas muestras en relación a los parámetros a entrenar?”) ya lo podemos responder con estos últimos dos ejercicios, si bien impactan la cantidad de muestras no es tan crítico en este práctico.

Faltó realizar una exploración random y ejecutar cada modelo varias veces con distintas inicializaciones, luego evaluarlos en test y poder quedarse con el modelo de menor variación en su accuracy.

 # Repositorio

https://github.com/argbat/deeplearning