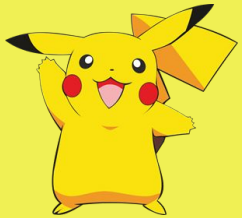
Trabalho Final

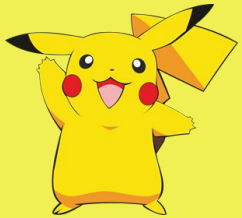
Classificação de pokémon

Introdução



- Neste trabalho, optamos por criar um modelo que visa identificar alguns tipos de pokémon.
- Escolhemos esse tema por ser algo bastante conhecido e interessante
- Adotamos 11 classes:
  - Pikachu
  - Charmander
  - Squirtle
  - Bubassauro
  - Articuno
  - Buttlerfree
  - Dragonite
  - Gengar
  - Nidoking
  - Magikarp
  - MewTwo

Metodologia



- Conjunto de dados balanceados, cada classe contando com 50 imagens, das quais 70% foram dedicadas para treino e 30% para teste
- Utilizamos o backend keras para definir o modelo e o frontend tensorflow para definição de pipelines de dataset
- Foi utilizada a técnica de augumentação para ampliar nossos dados de treino, aplicando filtros como contraste e brilho a cada época, diminuindo as chances do modelo decorar a base, aumentando a variabilidade dos dados
- Escolhemos o otimizador Adam com algumas callbacks
    - Learning rate dinâmico
    - Early stop

Código

```
return tf.keras.Sequential([
    tf.keras.layers.Conv2D(64, (5, 5), padding='same', input_shape=(128, 128, 3), activation='relu'
    tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
    tf.keras.layers.Conv2D(128, (3, 3), padding='same', activation='relu'),
    tf.keras.layers.MaxPooling2D(pool_size = (2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(AMOUNT_CLASSES, activation='softmax')
])
```

Código

```
In [118]: model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)            (None, 128, 128, 64)      4864

 max_pooling2d_4 (MaxPooling  (None, 64, 64, 64)       0
 2D)

 conv2d_5 (Conv2D)            (None, 64, 64, 128)       73856

 max_pooling2d_5 (MaxPooling  (None, 32, 32, 128)      0
 2D)

 flatten_2 (Flatten)          (None, 131072)            0

 dense_2 (Dense)              (None, 11)                1441803

=================================================================
Total params: 1,520,523
Trainable params: 1,520,523
Non-trainable params: 0
_________________________________________________________________
```

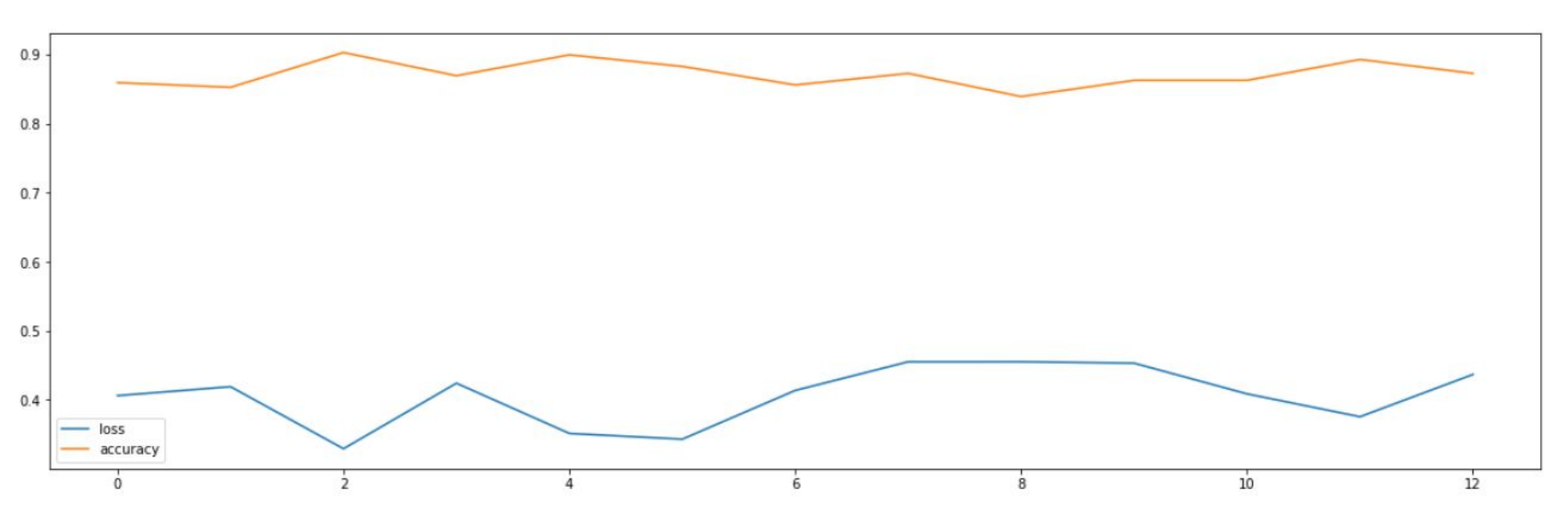
Resultados

- Ao final de 32 épocas (em duas runs), conseguimos algumas métricas interessantes em treino
  - Loss: 0.4367
  - Accuracy: 0.8725
- Ao aplicar o modelo na base de testes, chegamos a alguns resultados promissores

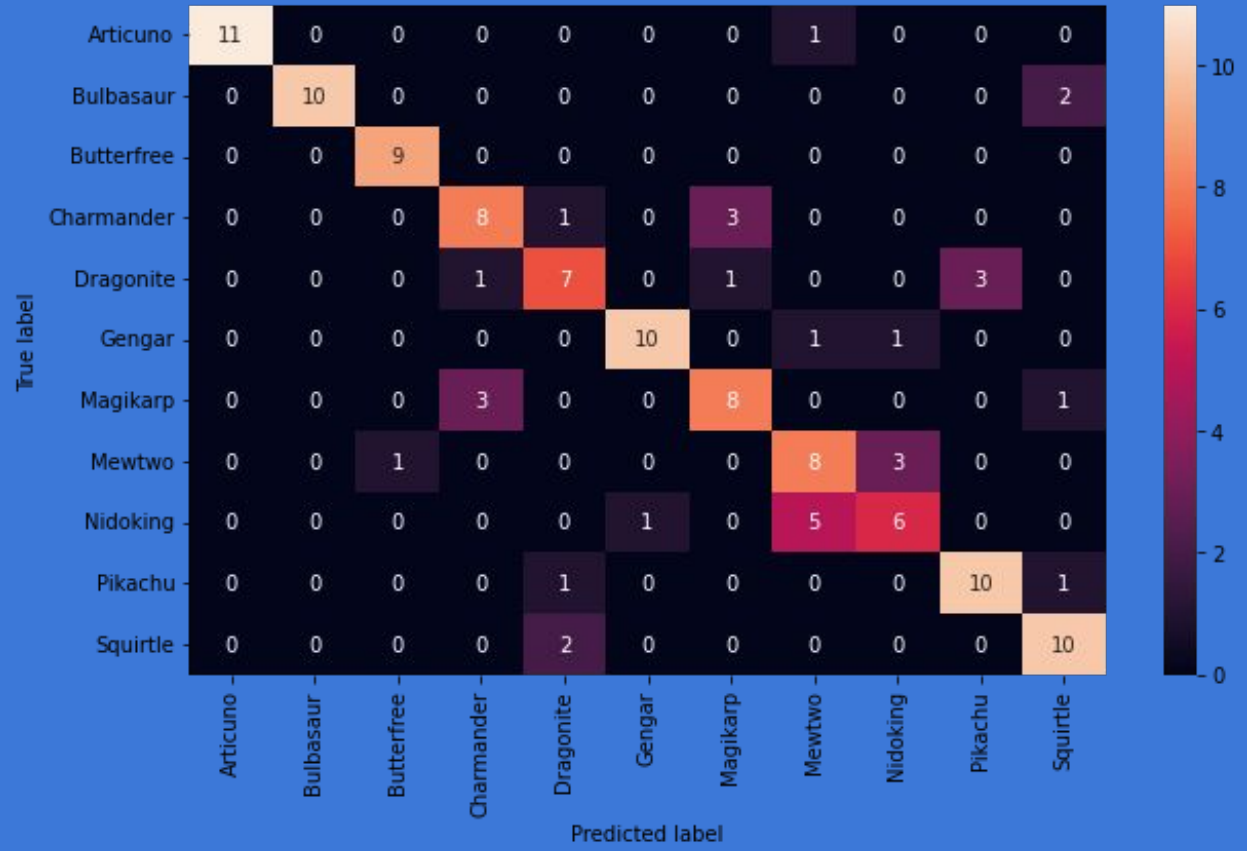
Resultados

	precision	recall	f1-score	support
0	1.00	0.92	0.96	12
1	1.00	0.83	0.91	12
2	0.90	1.00	0.95	9
3	0.67	0.67	0.67	12
4	0.64	0.58	0.61	12
5	0.91	0.83	0.87	12
6	0.67	0.67	0.67	12
7	0.53	0.67	0.59	12
8	0.60	0.50	0.55	12
9	0.77	0.83	0.80	12
10	0.71	0.83	0.77	12
accuracy			0.75	129
macro avg	0.76	0.76	0.76	129
weighted avg	0.76	0.75	0.75	129

Resultados



Matriz de Confusão



Erros Interessantes

X

Erros Interessantes

X

Erros Interessantes

Nidoking

X

Mewtwo

Conclusão

- Diante dos resultados apresentados, conseguimos observar que o modelo comporta bem, conseguindo categorizar de forma minimamente aceitável os pokémon
- Percebe-se que em casos específicos, como o pikachu com dragonite, ocorre alguns erros, por conta de certos padrões entre eles, como cor e forma
- Ao final, foi uma ótima forma de evoluir os aprendizados vistos em sala, colocando em prática conceitos de uma forma lúdica.