CLASSEZ DES IMAGES À L'AIDE D'ALGORITHMES DE DEEP LEARNING

OPENCLASSROOMS - INGÉNIEUR MACHINE LEARNING

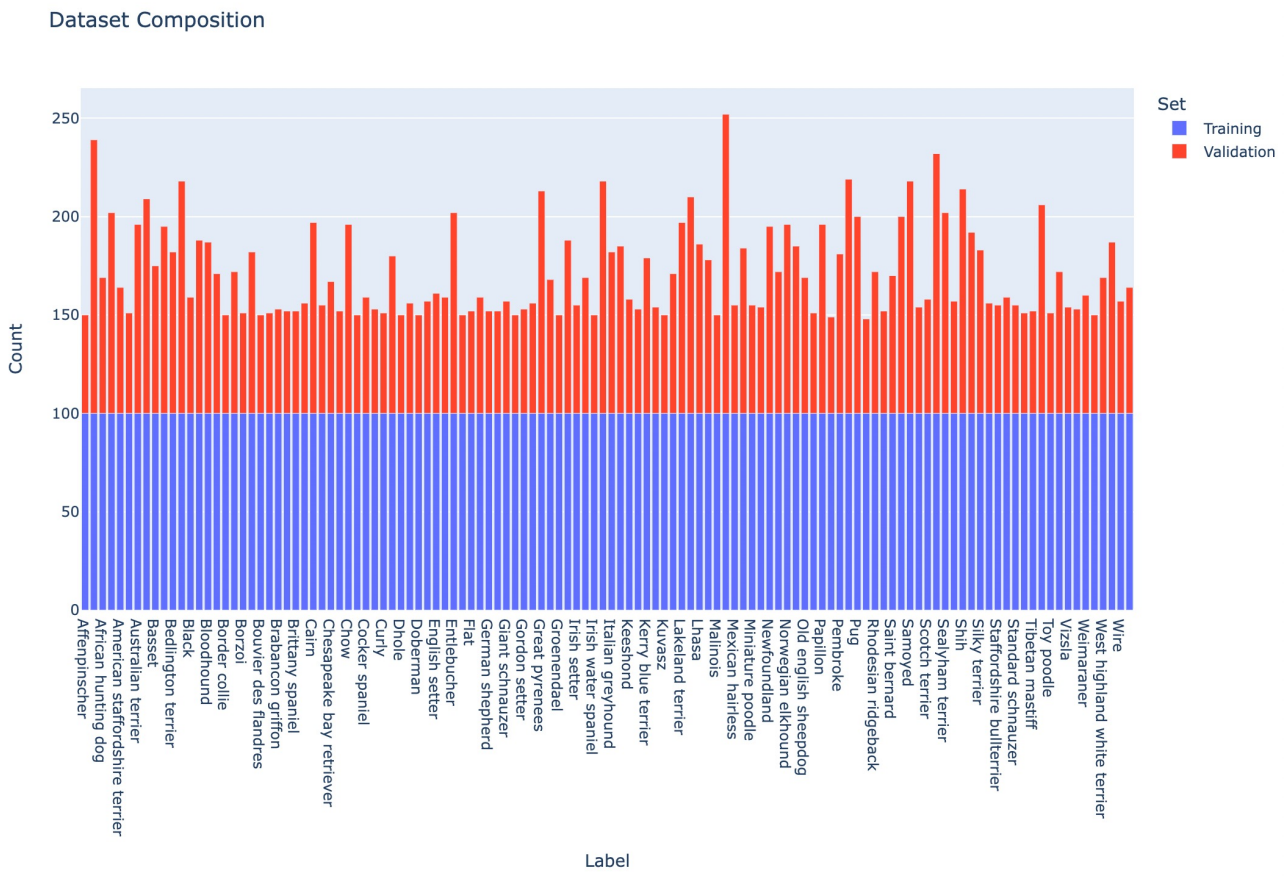
THIBAUD GROSJEAN - MARS 2022

PRÉSENTATION DU PROJET

- Mission :
  - Association de protection des animaux
  - Utilisation du dataset Standford Dogs
- Objectifs :
  - Entrainer un algorithme de détection des races de chiens
  - Développer une application qui fournit des prédictions
  - Créer un support de présentation

JEU DE DONNÉES

- Stanford Dogs
- Métadonnées :
  - Chemins vers les fichiers sources
  - Catégories
  - Annotations
- Taille : 757MB



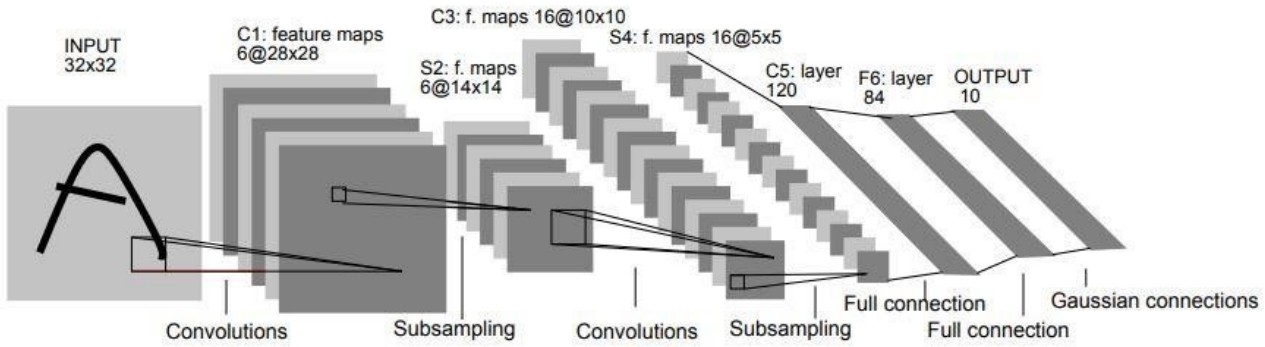
JEU DE DONNÉES

- Nombre de catégories : 120
- Nombre d'images : 20580
  - Train : 12000
  - Validation : 8580
- Le set de validation est déséquilibré

PRÉTRAITEMENT DES DONNÉES

- Filtrage du dataset
- Utilisation de *ImageDataGenerator (Keras)*
  - Ingestion
  - Performance (Batch)
  - Transformation
  - Data Augmentation

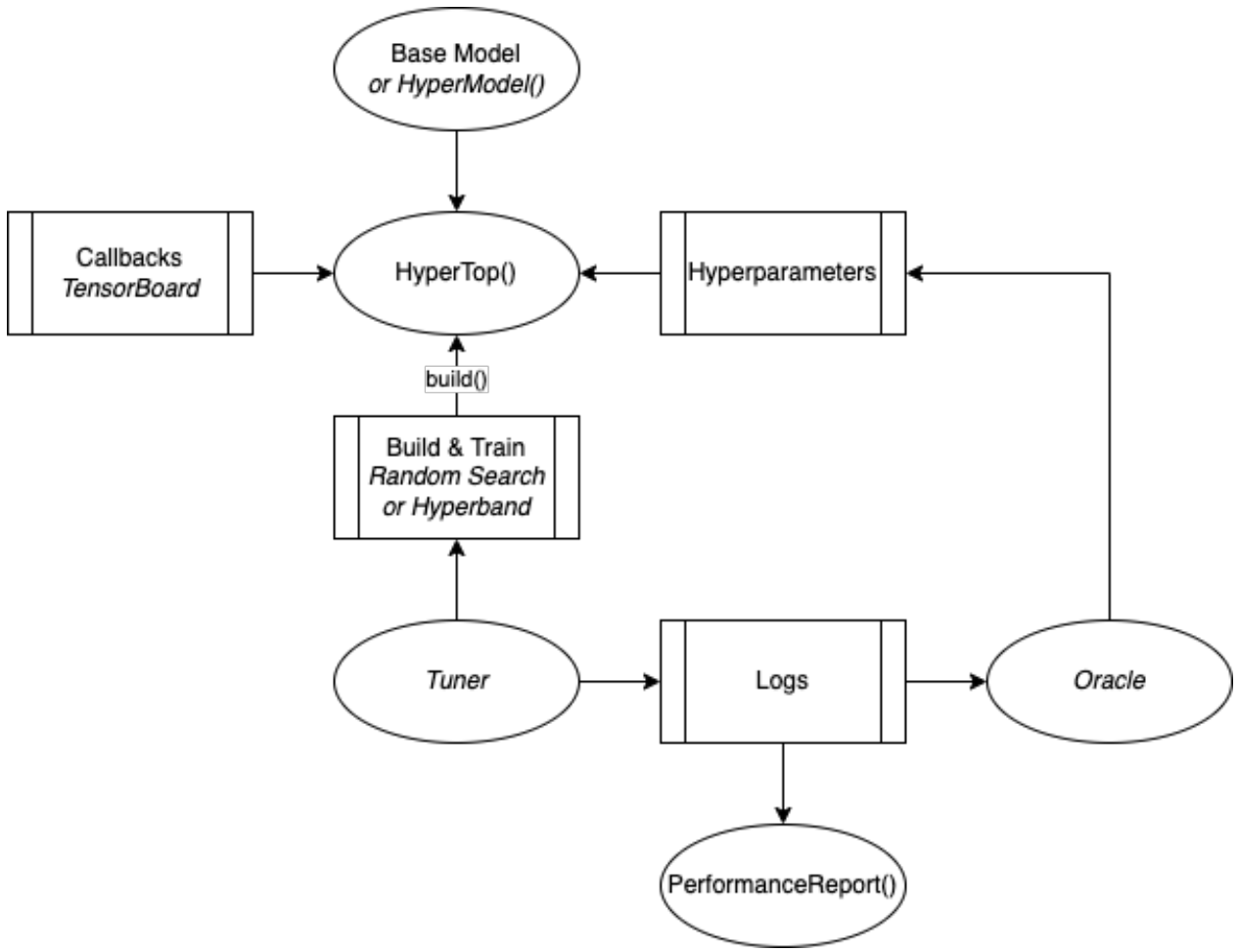
ALGORITHMES IMPLÉMENTÉS



RÉSEAU NEURONAL CONVOLUTIF

- *Input Layer*
- Convolutional Layer
- Pooling Layer
- Fully Connected Layer
- *Output Layer*

ARCHITECTURE DU FRAMEWORK DÉVELOPPÉ



- Keras *(API Séquentielle)*
- KerasTuner *(Tuning des hyperparamètres)*
- Tensorboard *(Evaluation des modèles)*

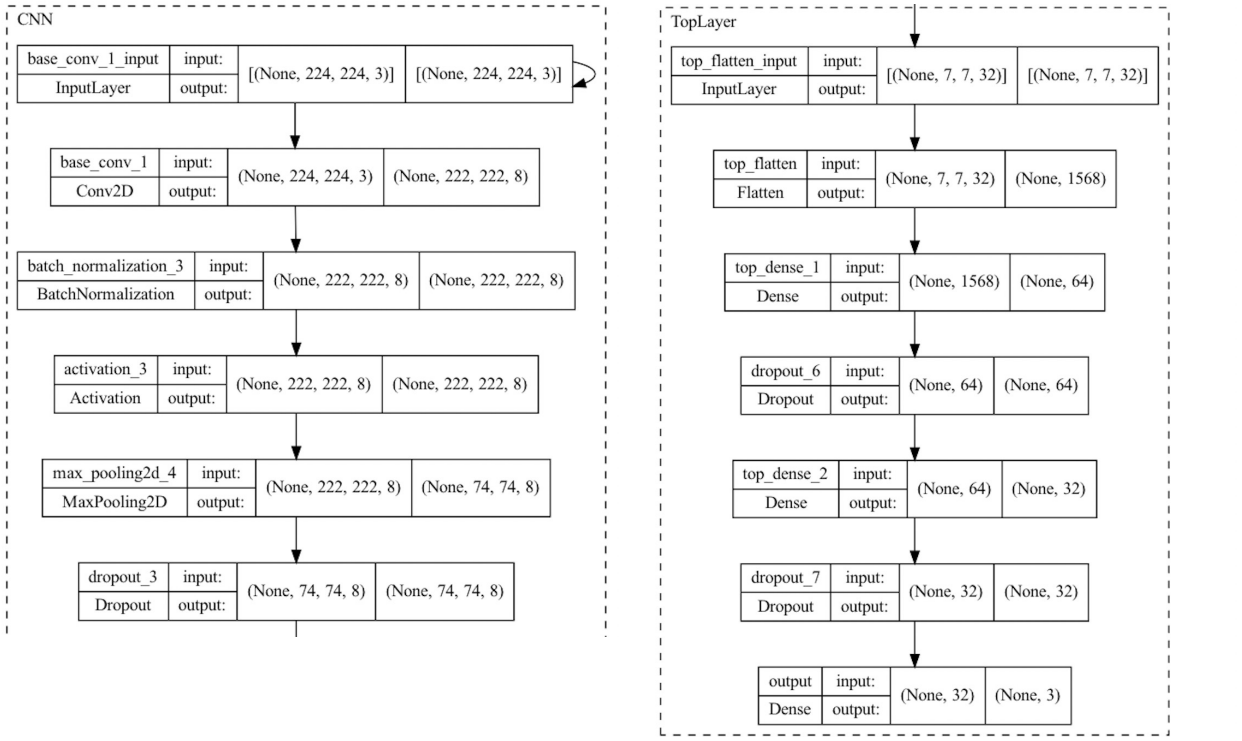
```python
class HyperTop(CustomHyperModel):

    def __init__(self, base_model, output_shape, *args, **kwargs):

        # Inits the instance with the default hyperparameters
        self.base_model = base_model
        self.params = dict(
            output_shape=output_shape,
            metrics=[
                'accuracy',
                tfa.metrics.F1Score(
                    num_classes=output_shape,
                    average='macro')])
        self.default_hyperparams = dict(
            drop_out=[0.],
            layer_1_units=[128],
            layer_2_units=[64],
            first_layers_activation=['relu'],
            output_activation=['softmax'],
            optimizer=['adam'],
            learning_rate = [1e-3])
        self.hyperparams =copy.deepcopy(self.default_hyperparams)
        self.final_model = None

    def build(self, hp=kt.HyperParameters()):

        # Get params
        params = self.params
        hyperparams = self.hyperparams

        first_layers_activation = hp.Choice(
                name='first_layers_activation',
                values=hyperparams.get('first_layers_activation'),
                default='relu')
        # Build the model if it is a KerasTuner hypermodel
        if isinstance(self.base_model, kt.HyperModel):
            base_model = self.base_model.build(hp)
            self.base_model = base_model
        base_model = self.base_model

        # Build the top layers
        model = models.Sequential(name='TopLayer')
        ## Flatten layer
```

GÉNÉRATION DE MODÈLES

- kt.CustomHyperModel()
- kt.HyperParameters()

GÉNÉRATION DE MODÈLES

- HyperCNN()
- HyperTop()



```
Trial 4 Complete [00h 02m 46s]
val_loss: 1.0303843021392822

Best val_loss So Far: 1.0123742818832397
Total elapsed time: 00h 10m 44s

Search: Running Trial #5

Hyperparameter    |Value             |Best Value So Far
first_layers_ac...|tanh              |relu
kernel_dim        |3                 |6
base_filters_1    |16                |8
base_filters_2    |16                |16
base_filters_3    |32                |128
drop_out          |0                 |0.3
layer_1_units     |32                |128
layer_2_units     |16                |16
layer_3_activation|sigmoid           |sigmoid
optimizer         |adam              |sgd
learning_rate     |0.001             |1e-05

Epoch 1/10

2022-05-02 17:33:48.616625: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is
enabled.

13/12 [==============================] - ETA: 0s - loss: 1.4208 - accuracy: 0.3467 - f1_score: 0.3686

2022-05-02 17:33:59.269098: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:113] Plugin optimizer for device_type GPU is
enabled.

12/12 [==============================] - 19s 2s/step - loss: 1.4208 - accuracy: 0.3467 - f1_score: 0.3686 - val_loss: 1.2678 - val_accuracy:
0.3772 - val_f1_score: 0.2806
Epoch 2/10
12/12 [==============================] - 16s 1s/step - loss: 1.3278 - accuracy: 0.3600 - f1_score: 0.3087 - val_loss: 1.3230 - val_accuracy:
0.2007 - val_f1_score: 0.1474
Epoch 3/10
```

OPTIMISATION DES HYPERPARAMÈTRES

- HyperSeach()
  - Random Search
  - Hyperband
- Objectif : validation loss
- Callbacks

OPTIMISATION DES HYPERPARAMÈTRES

- Hyperparameter Space

```
{'base_filters_1': [8, 16, 32],
 'base_filters_2': [16, 32, 64],
 'base_filters_3': [32, 64, 128],
 'kernel_dim': [3, 6]}
```

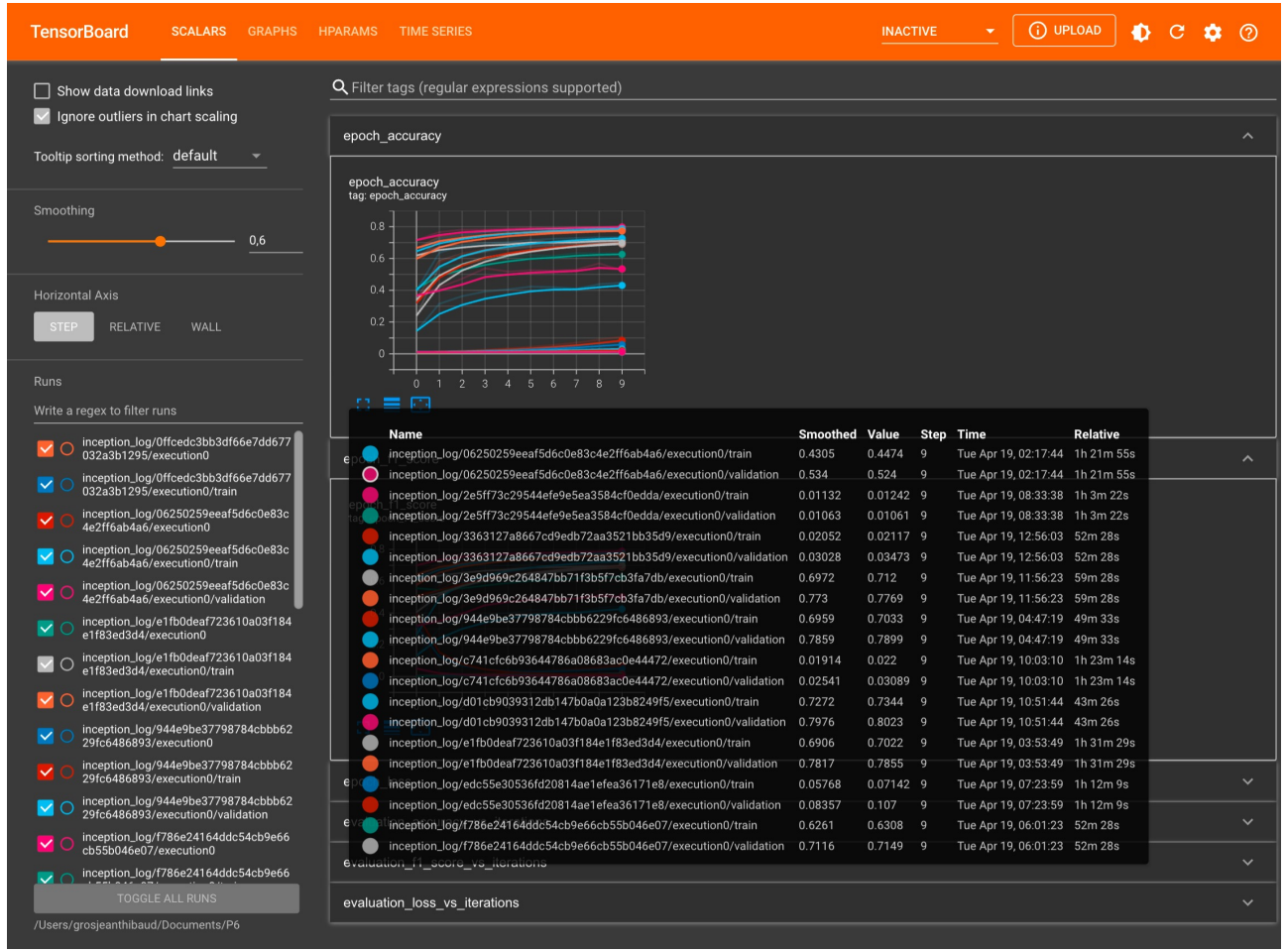
```
hypermodel.update_hyperparams(**hypertop_hyperparams)
hypermodel.hyperparams
```

[46] ✓ 0.6s

```
{'drop_out': [0.0, 0.6, 0.3],
 'layer_1_units': [128, 32, 64],
 'layer_2_units': [64, 16, 32],
 'first_layers_activation': ['tanh', 'relu'],
 'output_activation': ['sigmoid', 'softmax'],
 'optimizer': ['adam', 'sgd'],
 'learning_rate': [1e-05, 0.001]}
```

EVALUATION DES PERFORMANCES

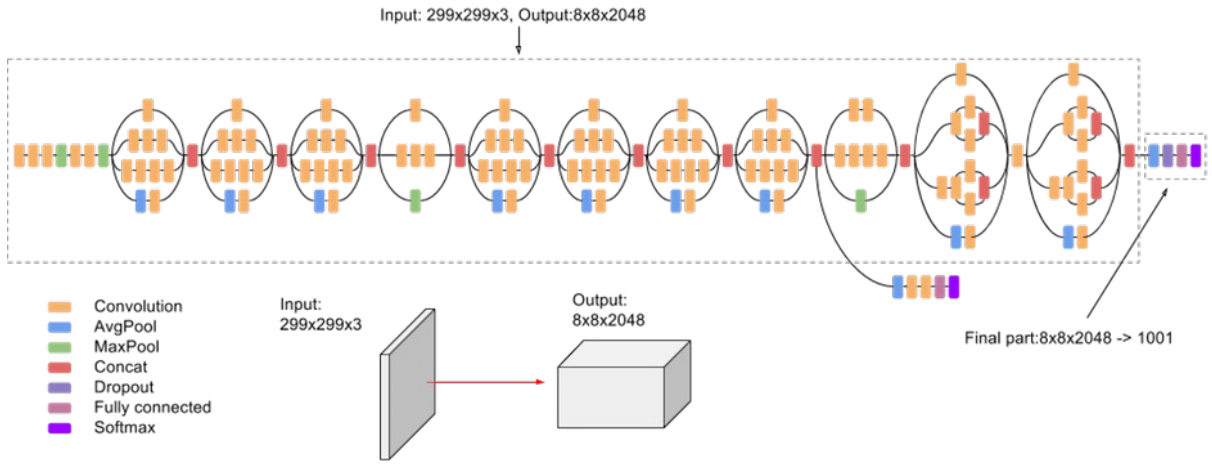
- Tensorboard



EVALUATION DES PERFORMANCES

algorithm	search	n_classes	train_score	val_score	val_accuracy	val_auc
cnn	random_search	3	0.549178	0.754739	0.564444	0.796504
vgg	random_search	3	0.089822	0.344714	0.777778	0.923002
inception	random_search	3	0.164762	0.0	1.0	0.99883
inception_120	random_search	120	1.118594	0.776255	0.726985	0.980366

- PerformanceReport()
  - Categorical Accuracy
  - AUC



APPLICATION

- Inception V3 (120 classes)
- Streamlit

CONCLUSION & PISTES D'AMÉLIORATION

- Modélisation
- Recherche d'hyperparamètres
- Evaluation
- Selection du modèle
- App (POC)

ÉCHANGE & QUESTIONS

Merci de votre attention !