TX - MÉTHODES D'APPRENTISSAGE PROFOND POUR L'IDENTIFICATION DES RACES DE CHIENS

Éloïse MOREIRA
eloise.moreira@etu.utc.fr

Damien DIEUDONNÉ
damien.dieudonne@etu.utc.fr

RÉSUMÉ

L'objectif est de développer une application iOS qui utilise l'appareil photo de l'iPhone pour photographier un chien et, si elle en contient effectivement un, identifier sa race via cette application.

DoggyDex

CONTENTS

1 REMERCIEMENT

Nous tenons à remercier notre responsable de TX, M. Mokhtar Alaya, pour son rôle crucial dans la supervision de ce projet. Son expertise et ses conseils ont été précieux.

La contribution de M. Alaya a été particulièrement notable dans le processus de compilation des modèles de Deep Learning. Grâce à son aide, nous avons pu exécuter des modèles avec de grands nombres d'itérations.

Enfin, nous sommes reconnaissants envers l'UTC pour l'opportunité de travailler sur un projet aussi innovant et stimulant. Les défis rencontrés et les connaissances acquises tout au long de ce projet resteront avec nous dans nos futures entreprises.

2 INTRODUCTION

Dans un monde où la technologie et la passion pour les animaux se croisent, notre projet "DoggyDex" trouve sa place. Cette application iOS vise à combiner l'amour pour les chiens avec l'innovation technologique. L'objectif principal de ce projet était de se former et d'acquérir de nouvelles compétences dans des domaines de pointe tels que le Deep Learning, le développement d'interface utilisateur avec SwiftUI, et la conception graphique UX/UI avec Figma.

La genèse de "DoggyDex" repose sur deux piliers : une passion partagée pour les chiens et la disponibilité d'un dataset en ligne de races canines. Inspirés par l'existence d'un défi Kaggle de reconnaissance de chiens, nous avons décidé de relever le défi de créer une application qui non seulement identifie les races de chiens à partir de photos, mais le fait d'une manière ludique et intuitive. La décision d'utiliser SwiftUI pour le développement sur iOS a été motivée par un intérêt personnel pour les produits Apple et le désir d'explorer ce langage de programmation.

Notre vision pour "DoggyDex" était de créer une application à la fois éducative et divertissante, destinée au grand public. En intégrant des technologies avancées telles que le Deep Learning avec TensorFlow et une interface utilisateur soignée conçue avec SwiftUI et Figma, nous avions pour objectif de rendre l'expérience d'apprentissage sur les races de chiens à la fois facile et agréable. L'objectif final est ambitieux mais réalisable : publier "DoggyDex" sur l'AppStore à la fin du semestre.

L'objectif principal de cette TX est l'acquisition de nouvelles compétences et connaissances. N'étant initialement pas familiers avec Figma, SwiftUI ni TensorFlow, il a été nécessaire de suivre une phase de formation approfondie et structurée. Les responsabilités ont été judicieusement réparties pour optimiser le processus d'apprentissage et la mise en œuvre du projet. Éloïse s'est consacrée à la maîtrise de Figma, avec pour mission de concevoir une maquette fonctionnelle et esthétiquement attrayante pour l'application. Damien, quant à lui, s'est immergé dans SwiftUI avec pour objectif de développer une application iOS, basée sur la maquette réalisée par Éloïse, garantissant ainsi une cohérence visuelle et fonctionnelle. Ensemble, nous avons exploré et approfondi TensorFlow, dans le but de développer un modèle de Deep Learning performant et précis pour la reconnaissance des races de chiens. Cette répartition des tâches a favorisé une approche collaborative et multidisciplinaire, essentielle à la réussite de notre projet.

Cette application représente non seulement une aventure dans l'apprentissage et l'application de compétences techniques avancées, mais aussi une opportunité d'apporter une contribution significative au monde des applications mobiles. À travers ce rapport, nous détaillerons le processus de développement de "DoggyDex", depuis la conception initiale jusqu'aux résultats finaux, en mettant en lumière les défis rencontrés, les solutions apportées, et l'expérience acquise tout au long de ce parcours innovant.

3 Développement des modèles Deep Learning

Pour le développement de l'application DoggyDex, deux modèles Deep Learning sont nécessaires : un modèle binaire qui détecte la présence ou non d'un chien sur l'image, puis si l'image contient un chien, un modèle qui détermine parmi 120 races possibles la race du chien photographié.

3.1 Formation DeepLearning et TensorFlow

Notre parcours dans le Deep Learning a commencé avec une base solide acquise lors de l'UV AI28 à l'UTC, où nous avons exploré les principes du Machine Learning. Cette expérience initiale a posé les fondations nécessaires pour notre incursion dans des concepts plus avancés.

Nous avons ensuite approfondi nos connaissances en Deep Learning en utilisant des ressources en ligne. La chaîne YouTube MachineLearnia[1] a été particulièrement utile, avec ses tutoriels guidant la création manuelle d'un réseau de neurones. Cet exercice pratique a renforcé notre compréhension des réseaux neuronaux et de leur apprentissage.

Le deep learning est une branche du machine learning qui s'inspire du fonctionnement du cerveau humain pour résoudre des problèmes complexes. Il utilise des réseaux de neurones artificiels pour apprendre à partir de données et effectuer des tâches spécifiques. On obtient en sortie des modèles qui nous permettent d effectuer des prédictions ou des détections comme pour notre projet, où les modèles doivent détecter la présence de chiens et leur race.

Les neurones artificiels, unités de base du deep learning, sont des éléments de traitement de l'information qui imitent les neurones biologiques : Chaque neurone reçoit des entrées, les pondère, les somme, puis applique une fonction d'activation pour produire une sortie.

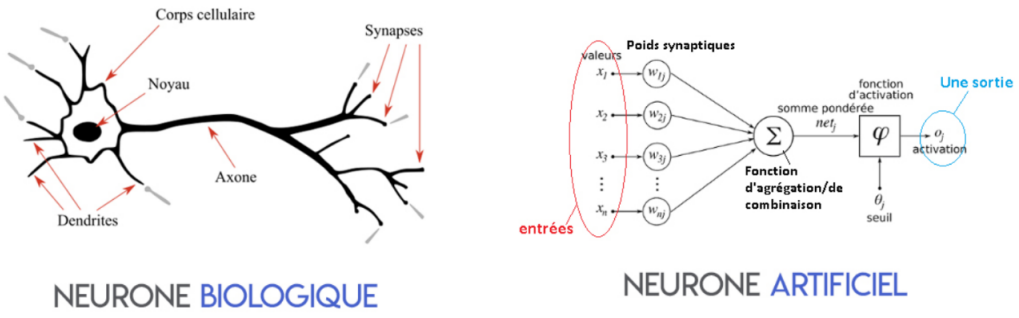

Figure 1: Neurone biologique et neurone artificiel

[1] Chaîne youtube MachineLearnia, Guillaume Saint-Cirgue, `https://www.youtube.com/c/MachineLearnia`

Dans notre quête de maîtrise du Deep Learning pour le projet DoggyDex, le document "Review of Deep Learning Algorithms and Architectures"[2] a joué un rôle crucial. Ce document nous a offert une compréhension approfondie des réseaux de neurones, *Neural Network* en anglais (NN), et de leur application dans divers domaines, notamment la vision par ordinateur et le traitement du langage naturel.

Le NN, une technique d'apprentissage machine inspirée du système nerveux humain, comprend des unités de traitement organisées en couches d'entrée, cachées et de sortie. La force de ces réseaux réside dans leur capacité à apprendre des représentations hiérarchiques des données. Cette caractéristique est particulièrement pertinente pour notre projet, où la reconnaissance précise des races de chiens à partir d'images est essentielle.

L'accent mis sur les réseaux de neurones convolutifs, *Convolutional neural neural network* en anglais (CNN), dans le document a été d'une importance capitale. Les CNN, en particulier, sont adaptés à l'apprentissage à partir d'images. Ils traitent les données d'entrée à travers des couches successives, extrayant progressivement des caractéristiques de niveau supérieur. Cette capacité à apprendre des représentations complexes à partir de données brutes est au cœur de notre approche pour identifier la présence de chiens et les races de chiens.

Le document détaille également les fonctions d'activation telles que la fonction ReLU, qui est essentielle pour introduire la non-linéarité dans le modèle, permettant ainsi au réseau d'apprendre des motifs plus complexes. Nous avons directement appliqué ce concept dans notre modèle CNN pour DoggyDex, en utilisant la fonction ReLU pour améliorer la capacité d'apprentissage du réseau.

En outre, le document explique le processus global de mise en œuvre des réseaux neuronaux, qui comprend l'acquisition de données d'entraînement et de test, l'entraînement du réseau et la réalisation de prédictions avec les données de test. Cette structure a guidé notre approche dans le développement du modèle de Deep Learning pour DoggyDex, en nous aidant à structurer efficacement notre processus d'apprentissage et de test.

La lecture de ce document a non seulement renforcé notre compréhension théorique du Deep Learning, mais a également fourni des perspectives pratiques sur la manière d'appliquer ces concepts à notre projet. Elle a été une étape clé dans notre formation, nous permettant de développer un modèle CNN robuste et efficace pour la reconnaissance de la présence d'un chien sur une photographie et des races de chiens dans DoggyDex.

La série de tutoriels sur TensorFlow proposée par Thibault Neveu[3] sur YouTube a marqué une étape décisive dans notre formation. Ces vidéos nous ont introduits à l'utilisation pratique de TensorFlow, notamment pour des projets de reconnaissance d'images, comme la reconnaissance de vêtements. Ces connaissances ont été directement applicables à notre projet de reconnaissance de races de chiens.

[2] Review of Deep Learning Algorithms and Architectures, `https://ieeexplore.ieee.org/document/8694781`

[3] Chaîne youtube ThibaultNeveu, `https://www.youtube.com/@ThibaultNeveu/`

La mise en application de ces connaissances a nécessité la création d'un modèle de base de réseau de neurones avec TensorFlow et Keras. Ce modèle a été ensuite adapté spécifiquement pour les besoins du DoggyDex. La consultation approfondie de la documentation TensorFlow a été un élément clé pour adapter les concepts de base à notre application spécifique.

Ce processus de formation nous a préparés techniquement, tout en stimulant notre créativité et notre capacité à résoudre des problèmes, des compétences essentielles pour le développement réussi de DoggyDex.

3.1.1 IMPLÉMENTATION DES MODÈLES

- Augmentation des données
  Dans le cadre de notre projet DoggyDex, nous avons développé un modèle de réseau de neurones en utilisant la bibliothèque Keras et la fonction 'ImageDataGenerator' pour le prétraitement des données. Dans le cadre du deep learning, la préparation et l'augmentation des données sont essentielles pour améliorer la performance et la généralisation des modèles. La classe 'ImageDataGenerator' de Keras joue un rôle significatif dans ce processus, en particulier lorsqu'il s'agit de traiter des images. Le code présenté [1] illustre l'utilisation de 'ImageDataGenerator' pour la préparation des données d'image, incluant l'augmentation des données et la séparation des ensembles d'entraînement et de validation.
  Le ImageDataGenerator est initialisé avec des paramètres spécifiques pour l'augmentation des données. Le rescaling à '1./255' est une étape de normalisation cruciale, convertissant les valeurs de pixels en une échelle de 0 à 1, adaptée pour les modèles de deep learning. L'augmentation des données est réalisée à travers divers paramètres tels que rotation range, width shift range, height shift range, shear range, zoom range, et horizontal flip. On peut voir dans l'image suivante que ces modifications furent appliqué.

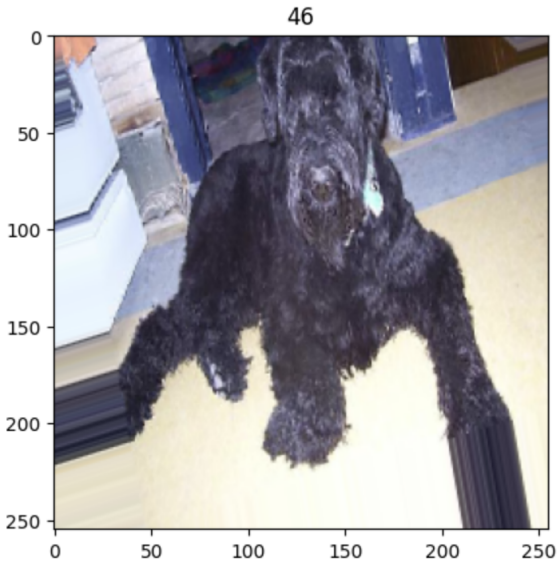

Figure 2: Exemple de modification d'une image

Ces transformations génèrent des variations des images originales, ce qui enrichit le jeu de données et aide à prévenir le surajustement.

Le mode de remplissage 'nearest' est utilisé pour les pixels nouvellement créés suite aux transformations. Un aspect crucial de ce setup est l'utilisation du 'validation split' à 0.2, qui réserve automatiquement 20% des données pour la validation.

Deux générateurs sont créés à partir de cette configuration [2](ToDo Annexe). Le 'train generator' sert à parcourir le répertoire spécifié pour l'entraînement, en appliquant les transformations définies et en produisant des lots d'images. Il utilise 80% des données disponibles dans le répertoire spécifié. En parallèle, le 'validation generator' utilise les 20% restants des données pour la validation. Cette séparation est essentielle pour évaluer la performance du modèle sur des données non vues pendant l'entraînement.

L'utilisation de 'ImageDataGenerator' dans ce contexte illustre sa flexibilité et son efficacité dans la préparation des données d'image pour le deep learning. Elle facilite non seulement l'augmentation des données mais aussi la gestion efficace de la séparation entre les données d'entraînement et de validation, se révélant ainsi être un outil indispensable pour l'entraînement de modèles d'apprentissage profond robustes et bien généralisés.

- Xception
  Le cœur de notre modèle reposait sur le modèle Xception pré-entraîné sur ImageNet. Ce choix a été motivé par la compatibilité de Xception avec notre jeu de données et par sa capacité à simplifier le fine-tuning.

  Ce modèle, représentant une évolution de l'architecture Inception, est basé sur le concept de convolutions séparables en profondeur, une technique qui décompose la convolution en deux parties : une convolution spatiale pour chaque canal d'entrée suivie d'une convolution ponctuelle.

  Ce qui distingue Xception, c'est son efficacité en termes de nombre de paramètres et de calculs nécessaires, tout en permettant à chaque couche d'apprendre des représentations plus spécifiques. L'architecture de Xception permet une meilleure séparation des caractéristiques apprises par les filtres du réseau, conduisant à une performance accrue sur des tâches complexes.

  Dans les projets de deep learning, l'utilisation de Xception avec des poids pré-entraînés, comme ceux obtenus à partir d'ImageNet, offre un avantage significatif. Cette technique, connue sous le nom de transfert d'apprentissage, permet d'utiliser des connaissances acquises sur un large ensemble de données pour des tâches spécifiques, ce qui est particulièrement bénéfique lorsqu'il y a une limitation des données d'entraînement. En outre, l'exclusion de la couche supérieure (‘include_top=False‘) lors de l'initialisation du modèle est une stratégie courante dans le transfert d'apprentissage, permettant l'ajout de couches personnalisées adaptées aux besoins spécifiques du projet.

  Ainsi, Xception se révèle être un choix privilégié pour les applications avancées de vision par ordinateur en raison de son architecture unique et de

ses performances impressionnantes. Sa flexibilité dans le cadre du transfert d'apprentissage le rend extrêmement adapté à une large gamme de tâches de deep learning.

- Optimiseur
  - Optimiseur Adam
    Dans le domaine de l'apprentissage automatique et plus particulièrement dans le contexte des réseaux de neurones, un optimiseur est un algorithme ou une méthode utilisée pour changer les attributs du modèle, tels que les poids des neurones, dans le but de réduire les pertes. L'optimisation est un processus clé pour permettre au modèle d'apprendre à partir des données et d'améliorer ses performances.
    La descente de gradient est l'une des techniques d'optimisation les plus fondamentales en apprentissage automatique. Elle est utilisée pour minimiser la fonction de perte (ou loss), qui est une mesure de l'écart entre la prédiction du modèle et la valeur réelle.
    Le gradient représente la pente de la fonction de perte et indique la direction dans laquelle la fonction augmente le plus rapidement. En calculant le gradient, on détermine dans quelle direction les poids doivent être ajustés pour réduire la perte. Mise à jour des Poids : Les poids sont ensuite ajustés dans la direction opposée au gradient. Cela signifie que si le gradient est positif (la perte augmente), les poids sont ajustés dans la direction négative (pour réduire la perte). Taux d'Apprentissage : Le taux d'apprentissage détermine la taille des pas faits dans la direction opposée au gradient. Un taux trop élevé peut conduire à des sauts trop importants, manquant le minimum de la fonction de perte, tandis qu'un taux trop faible rend l'apprentissage très lent.

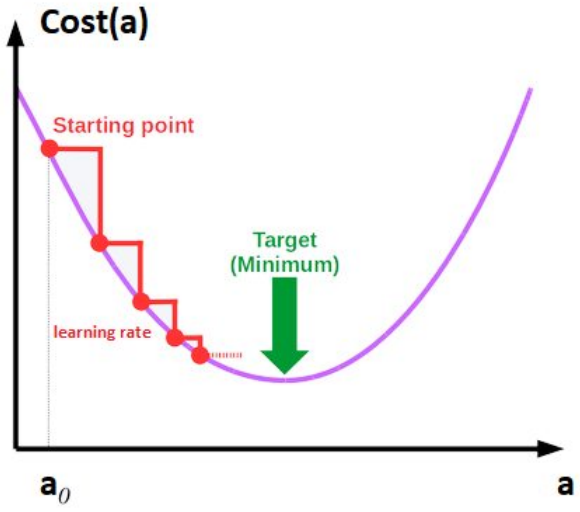

Figure 3: Gradient Descent Example

L'optimisur utilisé est Adam, qui signifie "Adaptive Moment Estimation". C'est un optimiseur qui ajuste le taux d'apprentissage de chaque paramètre individuellement. Il combine les avantages de deux

approches d'optimisation adaptatives, Momentum et RMSProp (Root Mean Square propagation).
L'algorithme ADAM ajuste les moments de premier ordre, qui représentent la moyenne mobile des gradients. La formule utilisée est :

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$$

où m_t est le moment de premier ordre au temps t, β_1 est le facteur de dépréciation pour le moment (généralement proche de 0.9), et g_t est le gradient à l'étape t. Mise à Jour des Moments de Second Ordre (Échelle des Gradients)
Les moments de second ordre sont également ajustés par ADAM. Ils sont définis comme la moyenne mobile des carrés des gradients :

$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$$

Ici, v_t est le moment de second ordre au temps t, et β_2 est le facteur de dépréciation pour ce moment (généralement proche de 0.999).
Correction de Biais pour les Moments
ADAM intègre une correction de biais pour les moments, ce qui est crucial surtout au début de l'entraînement lorsque t est petit. Les formules de correction sont :

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

Ces équations corrigent le biais initial vers zéro des moments m_t et v_t.
Mise à Jour des Poids
Finalement, ADAM met à jour les poids en utilisant l'équation suivante :

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t$$

où w_t représente les poids au temps t, α est le taux d'apprentissage, et ϵ est un petit nombre ajouté pour éviter la division par zéro (souvent autour de 10^{-8}).

- Taux d'apprentissage de 0,0001
  Un taux d'apprentissage finement choisi permet au modèle d'évoluer et de s'ajuster avec précision sans risquer de sauter des zones cruciales dans l'espace de la fonction de perte, un équilibre important pour la convergence efficace du modèle.

- Utilisation de CallBacks
  Ces callbacks ont été essentiels pour optimiser le processus d'entraînement, assurant que notre modèle apprenne efficacement tout en évitant le surapprentissage. Grâce à cette stratégie, nous avons pu atteindre une précision de 80% sur l'ensemble de validation, un résultat significatif qui témoigne de la robustesse et de l'efficacité de notre modèle dans la reconnaissance des races de chiens.

  - Early Stopping : Ce mécanisme interrompt l'entraînement si la perte de validation ne s'améliore pas après un certain nombre d'époques consécutives (dans notre cas, après 10 époques sans amélioration). L'early stopping est crucial pour prévenir le surapprentissage, car il empêche le modèle de continuer à apprendre des spécificités du jeu de données d'entraînement qui ne généralisent pas bien.
  - modèle Checkpoint : Ce callback sauvegarde le meilleur modèle en fonction de la perte de validation. À chaque fois qu'une amélioration est observée, le modèle est sauvegardé. Cela nous permet de récupérer la meilleure version du modèle à la fin de l'entraînement, indépendamment du moment où l'early stopping se déclenche.
  - ReduceLROnPlateau : Ce callback ajuste le taux d'apprentissage lorsque la perte de validation cesse de s'améliorer. En réduisant le taux d'apprentissage, nous avons pu affiner davantage les ajustements du modèle, permettant une meilleure convergence et une précision accrue.

- Hyperparamètre
  Dans le développement de notre modèle pour DoggyDex, deux hyperparamètres ont joué un rôle crucial : le dropout et le ReduceLROnPlateau. Ces hyperparamètres ont été soigneusement choisis et configurés pour optimiser la performance et la robustesse de notre modèle.

  - Dropout : Prévention du Surapprentissage
    Le dropout est une technique d'entraînement utilisée pour prévenir le surapprentissage dans les réseaux de neurones. Le surapprentissage se produit lorsque le modèle apprend des détails et du bruit spécifiques au jeu de données d'entraînement, au détriment de sa capacité à généraliser à de nouvelles données. Voici comment le dropout fonctionne :

    Pendant l'entraînement, le dropout "désactive" aléatoirement un certain pourcentage de neurones (ou de connexions entre les neurones) à chaque étape ou époque. Cela empêche le modèle de devenir trop dépendant de certains traits ou schémas dans les données d'entraînement, encourageant ainsi une meilleure généralisation. Nous avons intégré une couche de dropout dans notre architecture de réseau de neurones, avec un taux de dropout de 0.5. Cela signifie que la moitié des neurones sont aléatoirement désactivés à chaque étape d'entraînement, réduisant le risque de surapprentissage et favorisant une meilleure robustesse du modèle.
  - ReduceLROnPlateau : Optimisation du Taux d'Apprentissage Le ReduceLROnPlateau, mentionné précédemment dans la section sur les callbacks, est un autre hyperparamètre clé. Cette méthode ajuste dynamiquement le taux d'apprentissage en fonction de la performance du modèle :

    Lorsque l'entraînement atteint un plateau, c'est-à-dire que la performance du modèle cesse de s'améliorer, le ReduceLROnPlateau réduit le taux d'apprentissage. En abaissant le taux, le modèle peut effectuer

des ajustements plus fins dans l'espace des paramètres, ce qui peut aider à surmonter les plateaux de performance. Dans notre cas, les performances du modèle sont basé sur l'accuracy des données de validation.

Conformément à notre stratégie d'optimisation, le ReduceLROnPlateau a été configuré pour surveiller la perte de validation et ajuster le taux d'apprentissage si aucune amélioration n'était observée sur 5 d'époques.

- Pooling : Réduction de Dimensionnalité et de Complexité

  Le pooling est une opération couramment utilisée dans les CNN qui vise à réduire la dimension spatiale (la largeur et la hauteur) des cartes de caractéristiques (feature maps) obtenues après les couches convolutives. Cette réduction est réalisée sans perdre d'informations significatives sur les caractéristiques importantes. Les deux formes les plus courantes de pooling sont :

  – Max Pooling : Sélectionne la valeur maximale d'une région de la carte de caractéristiques.
  – Average Pooling : Calcule la moyenne des valeurs dans une région de la carte de caractéristiques.

  Dans notre cas, nous avons utilisé le Global Average Pooling(GAP), qui est une variante de l'average pooling où, au lieu de prendre la moyenne d'une région locale, on calcule la moyenne de l'ensemble de la carte de caractéristiques pour chaque canal.

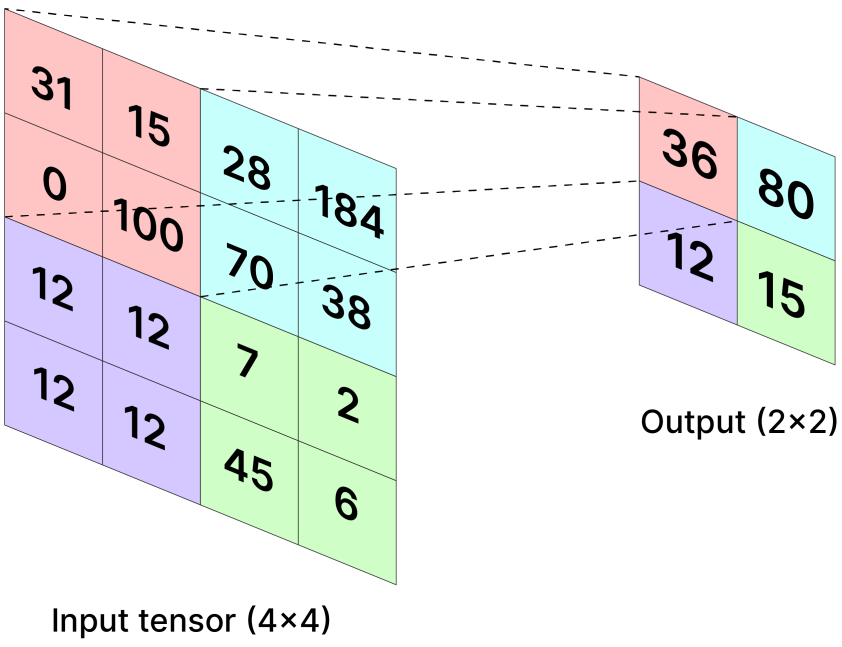

Figure 4: Average Pooling

Voici ses avantages et son application dans notre modèle :

Le GAP réduit chaque carte de caractéristiques à une seule valeur moyenne. Cela simplifie le modèle en réduisant le nombre total de paramètres et la

complexité computationnelle, ce qui est particulièrement utile pour prévenir le surapprentissage. Malgré la réduction drastique des dimensions, le GAP est capable de conserver les informations les plus cruciales présentes dans les cartes de caractéristiques. Cela est dû au fait qu'il capture l'essence globale des caractéristiques détectées par les couches convolutives. Dans DoggyDex, nous avons utilisé le GAP après les dernières couches convolutives de Xtension. Cette technique a permis de compresser les informations importantes avant de les transmettre aux couches denses suivantes, facilitant ainsi l'apprentissage des classifications finales des races de chiens.

- Couche Dense : Coeur du traitement des caractéristiques
  Dans notre modèle pour DoggyDex, après avoir réduit la dimensionnalité des données via le Global Average Pooling, nous avons utilisé des couches denses pour effectuer le traitement final des caractéristiques avant la classification. Les couches denses, également connues sous le nom de couches entièrement connectées, jouent un rôle crucial dans les réseaux de neurones pour la synthèse des données et la prise de décision.

  Une couche dense est une couche de réseau de neurones où chaque entrée est connectée à chaque sortie par un poids. Voici quelques points clés sur leur fonctionnement :

  – Connexions Complètes : Dans une couche dense, chaque neurone de la couche précédente est connecté à chaque neurone de la couche dense. Cela permet au réseau de traiter l'ensemble des caractéristiques extraites dans les couches précédentes.

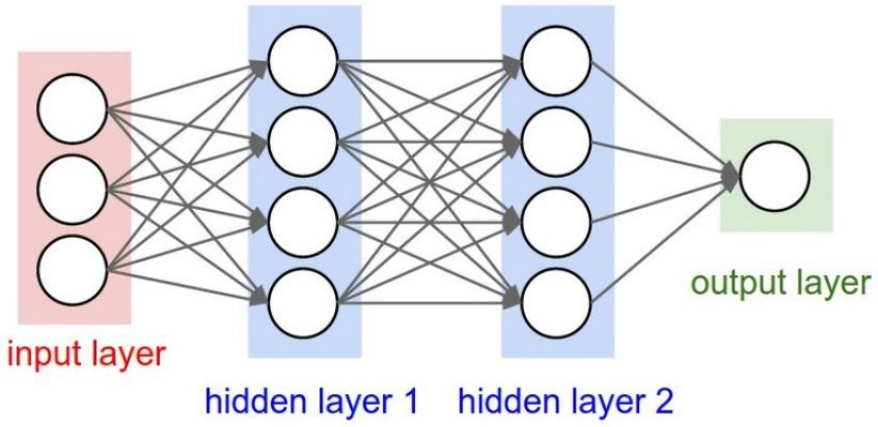

Figure 5: Dense Neural Network Example

  – Apprentissage des Caractéristiques de Haut Niveau : Alors que les couches convolutives et de pooling se concentrent sur l'extraction des caractéristiques locales et la réduction de dimensionnalité, les couches denses combinent ces caractéristiques pour apprendre des patterns de plus haut niveau, utiles pour la classification ou la prédiction.

  Dans le cadre de DoggyDex, les couches denses ont été déployées pour effectuer les tâches finales d'apprentissage et de classification . Après le Global Average Pooling, les caractéristiques réduites sont transmises aux

couches denses. Ici, le réseau apprend à combiner ces caractéristiques de manière significative pour faire des prédictions précises, dans notre cas, la classification des races de chiens.
Nous avons utilisé la fonction d'activation 'ReLU' dans les premières couches denses pour introduire la non-linéarité, permettant au modèle d'apprendre des relations complexes. Dans la dernière couche dense, la fonction d'activation 'Softmax' a été utilisée pour la classification multi-classes, produisant une distribution de probabilité sur les différentes races de chiens.

Les couches denses ont été essentielles pour notre modèle de reconnaissance des races de chiens. Elles ont permis de transformer les caractéristiques abstraites extraites par les couches convolutives et de pooling en décisions concrètes et justifiées, essentielles pour atteindre une précision élevée dans la classification.

Ainsi, après plusieurs tests nous avons deux couches Dense dans notre hiden layer à 1024 neurones chacune, et notre dernière couche, celle nécessaire à la prédiction, est de 120 neurones étant donné qu'il y a 120 races de chien à repérer.

- Metriques
  - Cross-Entropy Loss
    L'entropie croisée est une fonction de perte fondamentale pour les problèmes de classification multiclasse. Elle évalue à quel point la distribution des probabilités prédites par le modèle s'écarte de la distribution réelle. Dans le cas d'une classification multiclasse, la formule de l'entropie croisée est :

$$H(y, \hat{y}) = -\sum_{c=1}^{M} y_{o,c} \log(\hat{y}_{o,c})$$

    où M est le nombre total de classes, y est un vecteur binaire de taille M indiquant la classe réelle (avec 1 pour la classe réelle et 0 pour les autres), et $\hat{y}$ est le vecteur de probabilités prédites par le modèle pour chaque classe.
    Cette fonction de perte est particulièrement efficace pour les réseaux de neurones, car elle pénalise les erreurs en fonction de l'écart entre les probabilités prédites et les véritables étiquettes de classe.
  - Precision (Accuracy)

    L'accuracy est une mesure directe de la performance d'un modèle, définie comme la proportion de prédictions correctes parmi le total des prédictions :

$$\text{Accuracy} = \frac{\text{Nombre de prédictions correctes}}{\text{Nombre total de prédictions}}$$

Elle est particulièrement intuitive et facile à comprendre, mais peut être trompeuse dans des situations où les classes sont déséquilibrées.

Bien que l'entropie croisée et l'accuracy mesurent différents aspects de la performance d'un modèle, elles sont souvent utilisées ensemble pour évaluer et affiner les modèles de classification. L'entropie croisée est une métrique clé pendant l'entraînement pour optimiser les paramètres du modèle, tandis que l'accuracy est souvent utilisée comme une métrique d'évaluation pour juger de la performance globale du modèle.

Modèle 1 : Reconnaissance des races de chiens

Ce modèle est le principal de notre application, celui qui sert à reconnaitre les races de chiens. Il retourne une valeur entre 0 et 119, correspondant aux 120 races de chiens reconnaissable par le modèle.

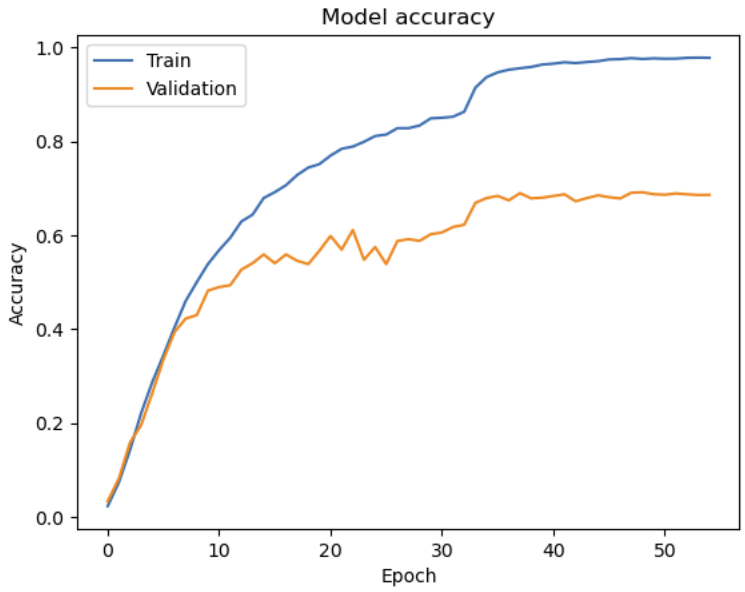

Figure 6: Train Accuracy

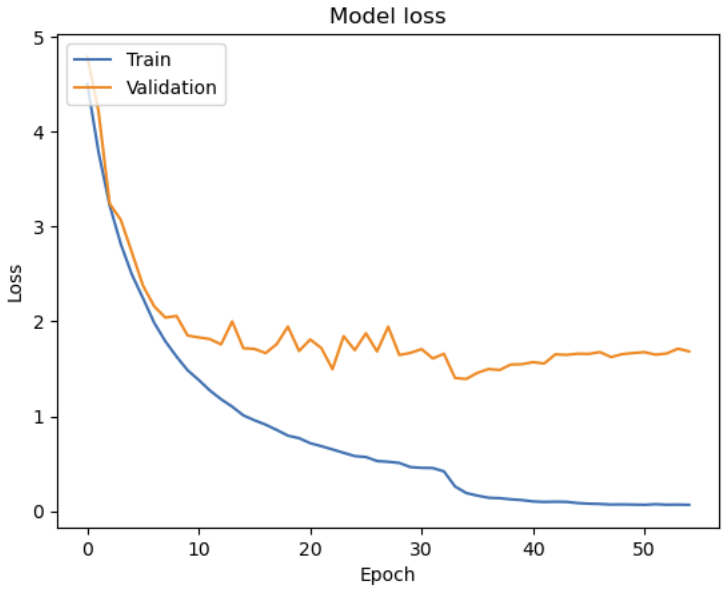

Figure 7: Train Loss

- Résultats de Précision du Modèle
  Sur une période de 50 époques, la précision du modèle sur l'ensemble d'entraînement a montré une amélioration continue, atteignant une valeur proche de 1 (100%). Cependant, la précision de l'ensemble de validation présente une croissance plus modeste et se stabilise autour de 0.6 (60%). La divergence croissante entre ces deux courbes suggère que le modèle s'adapte très bien aux données d'entraînement mais ne généralise pas de la même manière sur les données de validation, ce qui est un signe classique de surajustement.

- Analyse de la Perte du Modèle
  La courbe de perte pour l'ensemble d'entraînement montre une décroissance rapide qui se stabilise à mesure que le nombre d'époques augmente, suggérant que le modèle devient de plus en plus confiant dans ses prédictions pour ces données. En revanche, la courbe de perte de validation décroît également, mais avec des augmentations intermittentes suivies de réductions, ce qui reflète les ajustements du taux d'apprentissage par 'ReduceLROnPlateau'. Cette volatilité dans la courbe de perte de validation peut indiquer que le modèle bénéficie des ajustements du taux d'apprentissage pour affiner ses prédictions sur les données de validation.

- Discussion
  Les résultats illustrent l'efficacité du modèle à apprendre à partir de l'ensemble d'entraînement et soulignent la nécessité de techniques pour améliorer la généralisation sur les données de validation. L'utilisation de 'ReduceLROnPlateau' a eu un impact perceptible sur l'apprentissage, comme en témoignent les variations dans la courbe de perte de validation. Ces variations correspondent probablement aux points où le taux

d'apprentissage a été réduit, ce qui a permis des améliorations ultérieures dans la performance du modèle.

La différence marquée entre les précisions d'entraînement et de validation indique que, bien que le modèle ait appris les caractéristiques des données d'entraînement, il pourrait bénéficier de méthodes pour renforcer sa capacité à généraliser. Cela pourrait inclure l'augmentation des données, l'ajout de régularisation, ou l'exploration de différentes architectures de réseau neuronal.

En conclusion, le modèle a démontré une capacité d'apprentissage solide mais améliorable. Des étapes futures devraient se concentrer sur la réduction du surajustement et l'amélioration de la précision de validation pour garantir que le modèle est robuste et fiable lorsqu'il est confronté à de nouvelles images de races de chiens. Nous pouvons par contre noté que, si la race n'est pas le premier résultat du modèle, elle est au moins deuxième ou troisième, ce qui démontre tout de même l'efficacité de notre modèle. De plus, le modèle sauvegardé et utilisé est un modèle sauvegardé à l'epoch 35 grâce au callback CheckPoint, ainsi le modèle sauvegardé n'est pas un modèle avec trop de surajustement. Ce qui s'observe aussi dans les résultats des données de test.

```
Found 2110 images belonging to 120 classes.
66/66 [==============================] - 11s 149ms/step - loss: 1.8994 - accuracy: 0.6863
Accuracy sur le test set:  0.686255931854248
66/66 [==============================] - 8s 114ms/step
```

Figure 8: Résultat du set de test

On observe bien des résultats similaire à la validation, ce qui prouve que ce n'est pas un modèle surajusté.

Modèle 2 : Détection de la présence d'un chien sur l'image

Le deuxième modèle utilise la bibliothèque Keras pour créer, entraîner et évaluer un modèle de classification binaire d'images en utilisant l'architecture Xception pré-entraînée. Le modèle détecte la présence ou non de chiens sur la photographie fournie au modèle.

- Résultats de Précision du Modèle

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/xception/xception_weights_tf_dim_ordering_tf_kernels_notop.h5
83683744/83683744 [==============================] - 54s 1us/step
WARNING:tensorflow:`period` argument is deprecated. Please use `save_freq` to specify the frequency in number of batches seen.
C:\Users\emoreira\AppData\Local\Temp\ipykernel_22312\1601755728.py:37: UserWarning: `Model.fit_generator` is deprecated and will be removed in a future
  history = model.fit_generator(
1015/1014 [==============================] - ETA: -5s - loss: 0.0672 - accuracy: 0.9778
Epoch 1: val_loss improved from inf to 0.04666, saving model to test.h5
c:\Users\emoreira\AppData\Local\Programs\Python\Python310\lib\site-packages\keras\src\engine\training.py:3079: UserWarning: You are saving your model a
  saving_api.save_model(
1014/1014 [==============================] - 11005s 11s/step - loss: 0.0672 - accuracy: 0.9778 - val_loss: 0.0467 - val_accuracy: 0.9857 - lr: 1.0000e-
```

Figure 9: Resultats du modèle 2

L'accuracy est excellente (0.9778), cependant cela ne correspond pas à de l'overfitting car on veut voir que la val_accuracy (0.9857) est également très élevée.

- Test sur une base de données issue de Kaggle

```
Found 8828 images belonging to 2 classes.
276/276 [==============================] - 1128s 4s/step - loss: 0.0317 - accuracy: 0.9903
Accuracy sur le test set:  0.9902582764625549
276/276 [==============================] - 719s 3s/step
```

Figure 10: Résultat du test du modèle 2

Le résultat de classification binaire est plus que satisfaisant, avec une accuracy de 0.9903.

4 DÉVELOPPEMENT DE L'APPLICATION

Le développement d'une application mobile constitue un processus méthodique et complexe qui se découpe en 3 étapes fondamentales :

- La conception ;
- Le développement ;
- Le déploiement.

Une fois ces étapes complétées, l'optimisation et l'amélioration continue ainsi que la maintenance doivent être assurées pour le bon fonctionnement de l'application.

Dans le cadre de notre TX, nous nous sommes fixé l'objectif d'arriver à la phase de déploiement de notre application "DoggyDex".

4.1 CONCEPTION DE L'APPLICATION

La phase de conception occupe une place cruciale dans le développement d'une application mobile, jouant un rôle déterminant dans la réussite globale du projet. Pour évaluer la réussite d'un projet de développement d'application, il faut apprécier la réponse de cette application aux besoins client.

Cela nécessite en amont un travail sur l'identification des besoins :

- Comprendre les objectifs de l'application ;
- Définir les fonctionnalités principales ;
- Identifier le public cible.

Une fois que les besoins client ont été identifiés, une phase de conception qui traduit ces idées en plans concrets permet de poser des bases solides pour une application mobile réussie.

La phase de conception se divise en deux étapes essentielles :

- Élaboration de l'architecture de l'application répondant aux besoins client ;
- Création de wireframes (esquisse de conception) puis maquettes en support pour visualiser l'interface utilisateur.

4.1.1 IDENTIFICATION DES BESOINS CLIENT

L'identification des besoins client est fondamentale pour répondre aux attentes de celui-ci. La compréhension des besoins client permet de créer une proposition de valeur solide. Les caractéristiques du produit qui répondent spécifiquement aux besoins des clients contribuent à définir l'unicité du produit sur le marché.

Dans cette optique, Damien et Éloïse ont échangé sur les besoins à définir, constituant un fil rouge lors de la conception et du développement de l'application.

Les besoins suivants ont été définis :

- accès à la liste de races de chiens (120 d'après la base de données en date);
- distinction claire entre les races découvertes et non découvertes par l'utilisateur;
- détection de la race à partir de la photo prise par l'utilisateur de l'application;
- stockage des photographies prises si race de chien reconnue;
- capacité de détecter un chien ou non sur une photographie;
- accès aux informations à jour sur une race de chien découverte;
- possibilité de mettre en favori des races de chien;
- compatibilité avec la caméra frontale et arrière;
- personnalisation des préférences de l'interface (mode sombre, taille de police, ...);
- possibilité d'ajouter des races nouvelles à la base de données;
- base de données centralisée;
- interface attrayante et intuitive;
- adaptabilité au support (différents dispositifs tels qu'iPhone, iPad, Mac Book, ...).

Le public cible est le suivant : utilisateurs iOS intéressés ou passionnés de chiens souhaitant en savoir plus sur les races de chiens.

L'objectif de cette application est donc de proposer au public visé une une base de données centralisée et une interface attrayante et ergononomique, user-friendly.

4.1.2 ÉLABORATION D'UN WIREFRAME

Un wireframe est une représentation visuelle basique et schématique de l'interface utilisateur d'une application ou d'un site web. Il s'agit d'une esquisse simplifiée qui

met en évidence la disposition des éléments clés sans se préoccuper des détails visuels, des couleurs ou des polices spécifiques. Les wireframes sont souvent créés au début du processus de conception pour définir la structure et la disposition générale de l'interface.

Ils peuvent être réalisés à partir d'outils de conception ou sur papier. Pour cette application, le choix a été basé sur une réalisation avec tablette.

Après avoir identifié les besoins client, le wireframe suivant a été réalisé :

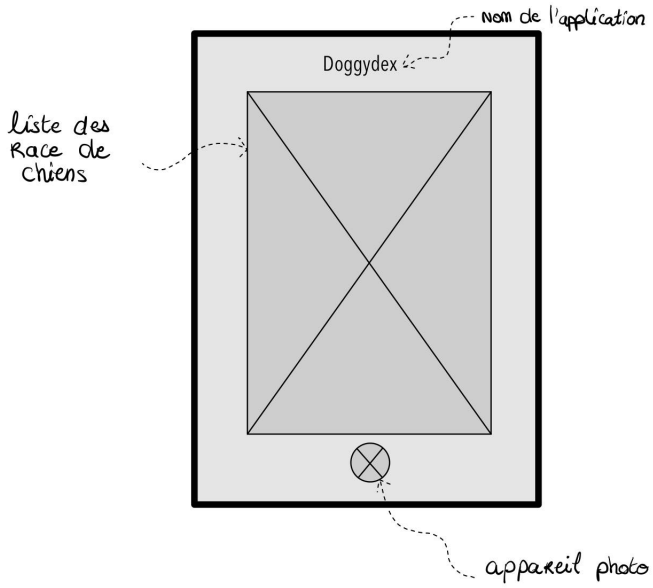

Figure 11: Wireframe - première page

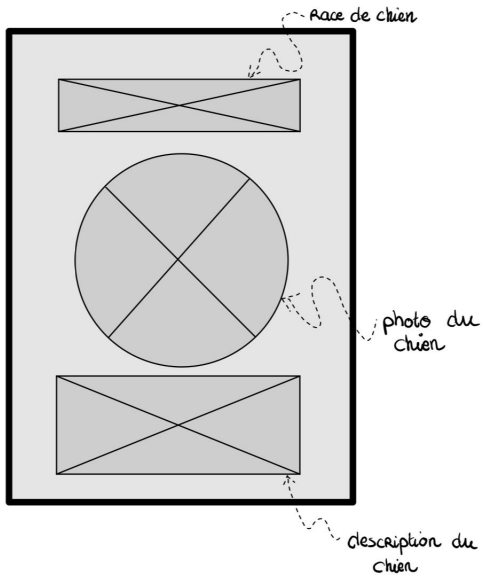

Figure 12: Wireframe - deuxième page

Ce wireframe permettent de résumer les besoins client essentiels auxquels l'application finale doit répondre pour une utilisation optimale.

4.1.3 FIGMA : L'OUTIL DE DESIGN UX/UI

Figma est une plateforme en ligne spécialisée dans la conception d'interfaces utilisateur. Elle offre des fonctionnalités avancées pour la création, le prototypage, et la collaboration en temps réel. Son utilisation est répandue dans le domaine du design d'expérience utilisateur (UX) et de l'interface utilisateur (UI), offrant une solution basée sur le cloud qui simplifie le processus de conception et favorise la collaboration entre les membres d'une équipe.

Voici une présentation des notions principales de Figma :

- Canvas :
  Le canvas représente l'espace de travail principal de Figma, l'arrière-plan sur lequel les éléments sont créés et organisés. C'est l'aire de jeu où les concepteurs agencent et assemblent les différents composants de leur projet.

- Frame :
  En UX/UI design, le concept central est celui de "frame". Un frame représente une zone délimitée dans laquelle le designer crée ses éléments. Il peut choisir le type de support, comme une interface pour un iPhone ou une tablette, en définissant les dimensions appropriées.

- Formes :
  Figma propose des outils pour créer une variété de formes géométriques, des rectangles aux cercles en passant par les polygones. Ces formes constituent les éléments de base pour construire une interface graphique.

- Texte :
  Figma permet d'ajouter du texte dans les designs, que ce soit pour des titres, des paragraphes ou des éléments d'interface utilisateur. Les options de personnalisation du texte incluent la police, la taille, la couleur, etc.

- Images Vectorielles ou Matricielles :
  Figma prend en charge à la fois les images vectorielles (scalables sans perte de qualité) et les images matricielles (images classiques composées de pixels). Cela offre une flexibilité pour intégrer différents types de contenus visuels.

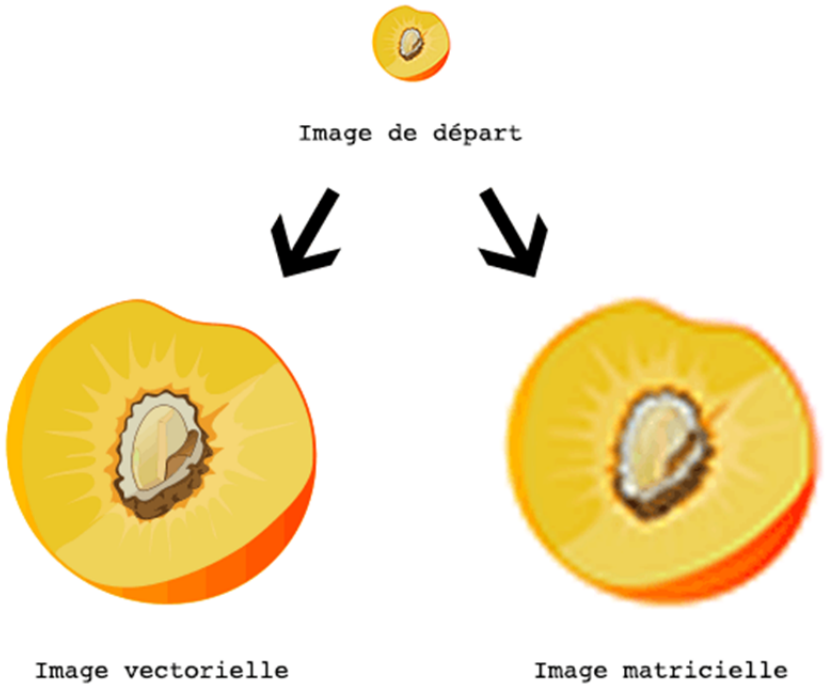

Figure 13: Images vectorielles et matricielles

Contrairement aux images matricielles (raster), qui sont composées de pixels, les images vectorielles utilisent des équations mathématiques pour décrire les contours et les formes.

- Composants Figma (Components) :
  Les composants sont des éléments réutilisables dans un design. Ils permettent de créer une bibliothèque d'éléments standardisés qui peuvent être facilement mis à jour et maintenus. Les changements apportés à un composant se reflètent automatiquement partout où il est utilisé.

- Grilles et Alignements :
  Figma offre des outils pour définir des grilles et assurer un alignement précis des éléments sur le canvas, favorisant la cohérence dans la mise en page.

- Effets et Ombres :
  Les concepteurs peuvent appliquer des effets spéciaux et des ombres pour donner de la profondeur et de la dimension aux éléments de l'interface.

Éloïse a pu apprendre à manipuler ces différentes notions grâce à une formation suivie en ligne proposée par Digidop [4], et ainsi réaliser les maquettes de l'application DoggyDex, présentées dans la partie suivante.

[4] Digidop, Agence de Développement Web No-Code, `https://academie.digidop.fr/formations/figma`

4.1.4 ÉLABORATION DE LA MAQUETTE DOGGYDEX

La phase de conception occupe une place cruciale dans le développement d'une application mobile, et élaborer une maquette pour la réalisation est fondamental, jouant un rôle déterminant dans la réussite globale du projet.

En effet, la maquette sert de fil rouge tout au long du projet pour le développement et pour la qualité. Pour le développement, une maquette bien élaborée sert de référence pour les développeurs lors de la mise en œuvre de l'application. Elle garantit une cohérence visuelle et fonctionnelle tout au long du processus de développement.
Elle est également importante pour la qualité : Pour évaluer la réussite globale d'un projet de développement d'application, il faut apprécier la réponse de cette application aux besoins client. En qualité, il faut comparer la conception au produit final et vérifier qu'il est bien conforme aux attentes définies dans la maquette.

Une première maquette, permettant à Éloïse de prendre en main l'outil Figma, a été réalisée. Voici le résultat :

Figure 14: Première maquette DoggyDex

Des images vectorielles ont été utilisées pour cette maquette : L'étoile est déjà designée comme forme sur Figma, mais les signets ont été créés à la main avec l'outil *pen*.

Figure 15: Images vectorielles de la première maquette

L'objectif étant de développer une application pour iOS, l'utilisation de templates accessibles sur Figma a été très utile et à pu ajouter de l'élangance aux maquettes.

Cela a permis notamment d'avoir un frame directement avec des dimensions d'iPhone et la barre d'information supérieure, pour une meilleure visualisation.

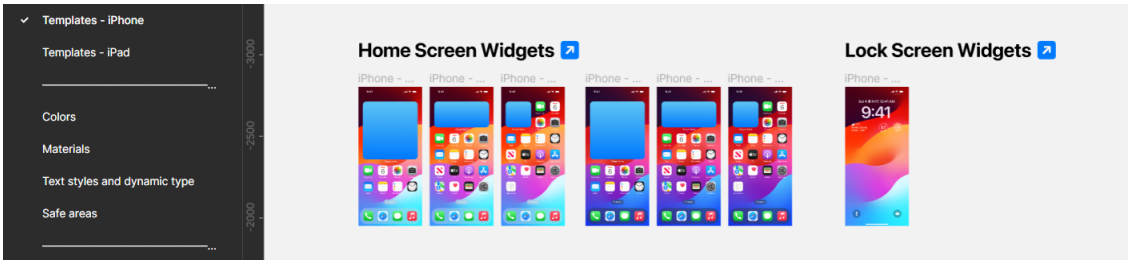

Figure 16: Apple design ressources figma

Bien que la maquette réponde aux attentes fonctionnelles du DoggyDex, celle-ci n'était pas esthétique et l'aspect visuel à dû être repensé.

Une veille UX UI a été réalisée sur Dribbble [5] et Behance [6] afin de s'inspirer sur les différents modèles UI possibles.

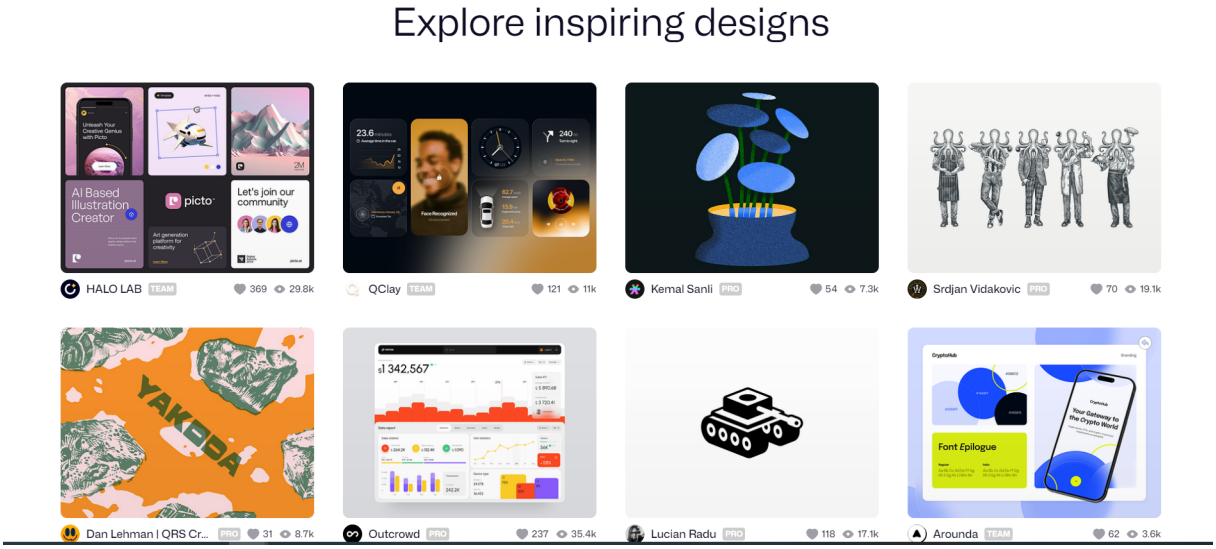

Figure 17: Site UI Dribbble

[5] Dribbble, `https://dribbble.com/`
[6] Behance, `https://www.behance.net/`

Adobe colors est également un outil très utile pour valider la conformité des couleurs [7].

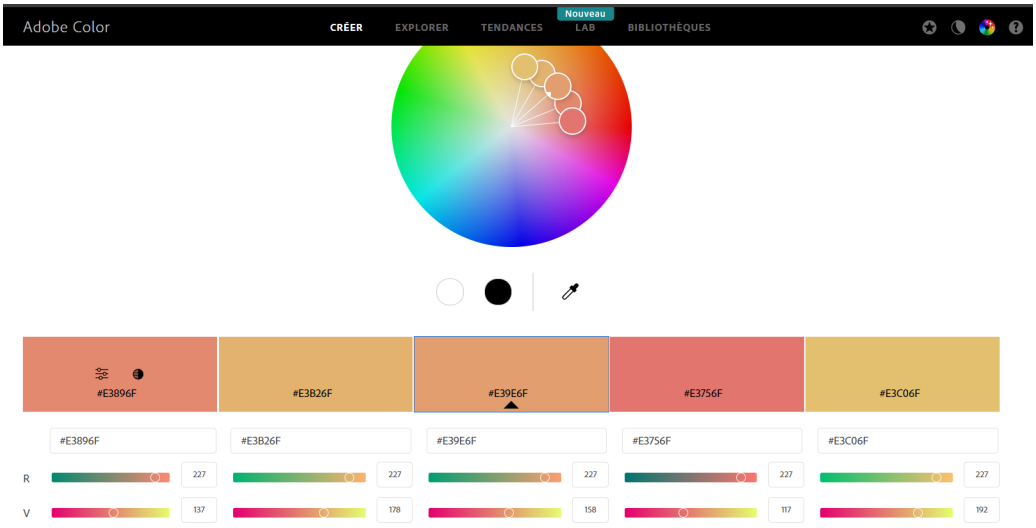

Figure 18: Adobe Colors

Avec la veille et Adobe colors, une deuxième maquette a été réalisée, répondant aux attentes esthétiques et fonctionnelles de l'application.

Figure 19: Deuxième maquette DoggyDex

Cette maquette a été le fil directeur du développement de l'interface UI avec SwiftUI.

[7] Adobe Colors, `https://color.adobe.com/fr/create/color-wheel`

4.2 DÉVELOPPMENT DE L'INTERFACE EN SWIFTUI

Suite à la conception initiale de notre maquette sur Figma pour l'application iOS, nous avons procédé au développement de cette application en utilisant SwiftUI avec Xcode. Cette approche nous a permis de mettre en œuvre efficacement toutes les fonctionnalités planifiées et de garantir une intégration parfaite avec l'écosystème iOS, aboutissant à une application pleinement fonctionnelle sur iPhone.

- Utilisation de Xcode dans le Développement
  Xcode, l'environnement de développement intégré (EDI) d'Apple, a joué un rôle crucial dans la réalisation de notre application de reconnaissance de races de chiens. En tant qu'outil de développement principal pour les applications iOS, Xcode nous a fourni une gamme complète de fonctionnalités pour concevoir, développer et déboguer notre application avec efficacité et précision.
  - Interface Intuitive et Outils Puissants
    Xcode offre une interface utilisateur intuitive qui facilite la visualisation et la manipulation des éléments de l'interface utilisateur de notre application. xCode se présente comme ceci :

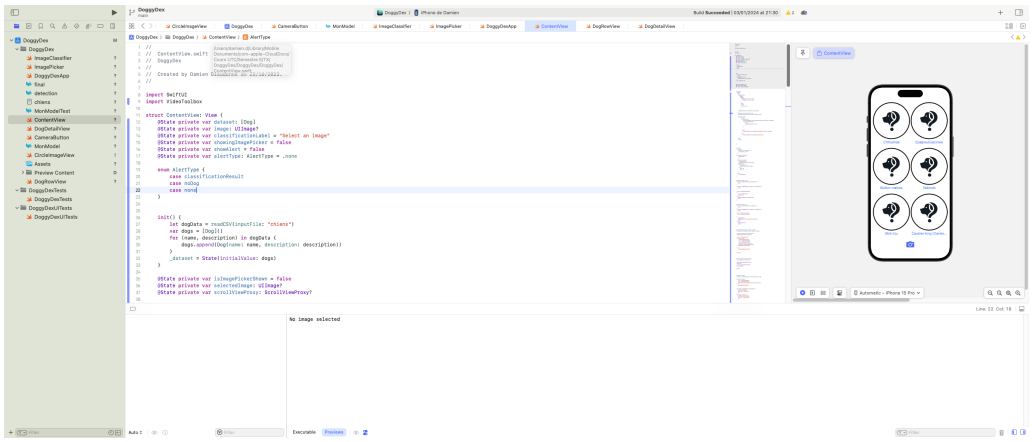

Figure 20: Application xCode

    Où a gauche nous retrouvons notre gestionnaire de fichier, au centre la zone de code et a droite une preview intéractive de notre application.
  - Intégration avec SwiftUI
    L'intégration transparente de Xcode avec SwiftUI a été un facteur déterminant dans le développement de notre application. L'aperçu en direct de SwiftUI, une fonctionnalité d'Xcode, nous a permis de visualiser en temps réel les modifications apportées à l'interface utilisateur, accélérant ainsi le processus de conception et de développement. Cette intégration a également simplifié la tâche de tester différentes configurations d'interface utilisateur pour divers appareils Apple.
  - Outils de Débogage et de Profilage
    Les outils de débogage avancés d'Xcode ont joué un rôle essentiel dans la résolution des problèmes et des bugs rencontrés lors du

développement. L'utilisation de l'instrument de profilage d'Xcode nous a permis d'analyser et d'optimiser les performances de notre application, en veillant à ce que le modèle de Deep Learning fonctionne de manière efficace et sans retard.

En résumé, Xcode s'est révélé être un outil indispensable dans le développement de notre application. Sa combinaison d'une interface utilisateur intuitive, d'une intégration poussée avec SwiftUI, d'outils de débogage puissants et de fonctionnalités de gestion de code a permis de mener à bien notre projet avec efficacité et innovation.

Bien sûr, passons à la section 3 de ton rapport, qui se concentre sur le développement de l'interface utilisateur avec SwiftUI :

- Développement de l'Interface Utilisateur avec SwiftUI**

  La création d'une interface utilisateur intuitive et réactive était un aspect essentiel de notre application. Cette section détaille comment nous avons utilisé SwiftUI pour concevoir et implémenter l'interface utilisateur de notre application.

  – Avantages de SwiftUI
  SwiftUI, avec sa syntaxe déclarative et son approche centrée sur les données. Sa capacité à s'adapter dynamiquement aux changements de données a rendu le développement de l'interface utilisateur fluide et efficace.

  – Conception de l'Interface avec SwiftUI
  L'interface de notre application a été conçue pour être à la fois esthétique et fonctionnelle. Nous avons utilisé divers éléments de SwiftUI, tels que des vues, des piles (stacks) et des listes, pour créer une expérience utilisateur harmonieuse. L'accent a été mis sur la facilité d'utilisation, en veillant à ce que les utilisateurs puissent naviguer intuitivement dans l'application et accéder facilement aux fonctionnalités.

  – Rôle du Fichier 'ContentView'
  Le fichier 'ContentView.swift' a servi de point central pour notre interface utilisateur. Ce fichier contient la vue principale de notre application, où les utilisateurs interagissent avec les différentes fonctionnalités. Nous avons structuré 'ContentView' de manière à refléter l'architecture globale de l'application, en intégrant le modèle de Deep Learning et en assurant une interaction fluide avec l'utilisateur.

  – Exemples de Composants d'Interface Utilisateur
  Nous avons intégré divers composants SwiftUI tels que des boutons pour la capture de photo, popups pour montrer les résultats de reconnaissance, et des images pour afficher les chiens identifiés. Chacun de ces éléments a été soigneusement conçu pour s'aligner avec l'esthétique globale de l'application, tout en fournissant une fonctionnalité claire et accessible.

  En conclusion, le développement de l'interface utilisateur avec SwiftUI a été un aspect crucial du succès de notre application. Grâce à la flexibilité et à la puissance de SwiftUI, nous avons pu créer une expérience utilisateur qui

non seulement répond aux besoins fonctionnels de l'application, mais le fait aussi d'une manière qui est visuellement attrayante et plaisante à utiliser.
En voici le résultat final :

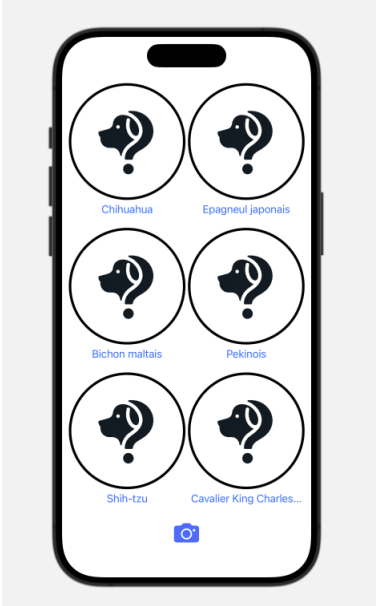

Figure 21: Page d'acceuil de l'application

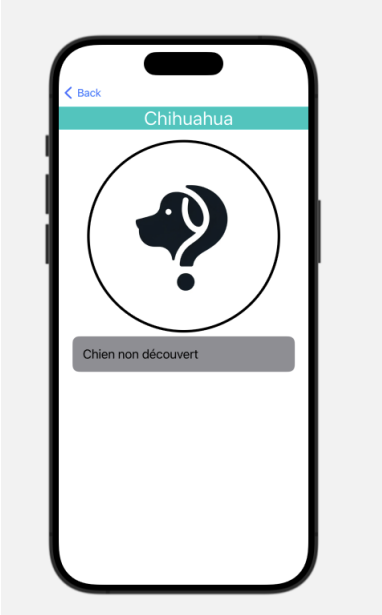

Figure 22: Page d'un chien

4.3 INTÉGRATRION

La capacité de notre application de reconnaissance des races de chiens à identifier correctement les différentes races repose sur un modèle de Deep Learning robuste. Pour intégrer ce modèle dans notre application développée avec Xcode, nous avons utilisé Core ML Tools, une bibliothèque python fourni par Apple. Cette section

décrit le processus d'intégration du modèle, de sa conversion à son implémentation dans Xcode.

- Utilisation de Core ML Tools pour la Conversion du Modèle
  Notre modèle de Deep Learning a été initialement développé et entraîné en utilisant des technologies de machine learning courantes, et il a été enregistré dans le format .h5. Pour l'intégrer dans notre application iOS, le format .h5 devait être converti en un format compatible avec Core ML. C'est ici que Core ML Tools entre en jeu. Cette bibliothèque permet de convertir des modèles de différents frameworks de machine learning, y compris Keras (utilisant le format .h5), en .mlmodel 1 ou .mlpackage, les formats pris en charge par Core ML, et donc Swift.
- Intégration avec SwiftUI
  L'intégration du modèle Core ML dans notre application SwiftUI nous permet d'utiliser directement le modèle de Deep Learning pour la reconnaissance des races de chiens. Lorsque l'utilisateur charge une image dans l'application, celle-ci est converti au bon format, donc le même format que les images d'entrainement, puis passée au modèle Core ML, qui effectue l'analyse et renvoie les résultats. Ces résultats sont enregistré dans une variable qui permet donc a l'iPhone de retourner le chien correspondant.

En résumé, l'utilisation de Core ML Tools pour convertir et intégrer notre modèle de Deep Learning dans Xcode a été une étape cruciale dans le développement de notre application. Cette méthode a non seulement simplifié le processus d'intégration, mais a également assuré que l'application peut exploiter pleinement la puissance du modèle de machine learning pour fournir des résultats précis et rapides.

5 CONCLUSION

Pour conclure, cette TX est une véritable réussite et nous à apporté beaucoup techniquement.

Damien a pu maîtriser le développement d'interface UI avec SwiftUI, ce qui constituera un atout pour sa carrière future, à laquelle il souhaite intégrer une expérience dans l'entreprise Apple.

Éloïse, qui était intéressée par la conception d'applications mobile et web, a pu avoir une première expérience avec un outil incoutournable du design UX/UI et poursuivra cet apprentissage en intégrant le design de maquettes au sein de projets professionnels.

Éloïse et Damien on également acquis une maîtrise des notions fondamentales du deep learning et du développement technique de modèles d'apprentissage profond, et le développement de l'application DoggyDex ouvre la voie à de futurs projets dans le machine et deep learning.

6 BIBLIOGRAPHIE

- MachineLearnia
  https://www.youtube.com/@MachineLearnia
- Thibault Neveu
  https://www.youtube.com/@ThibaultNeveu
- Review of Deep Learning Algorithms and Architectures https://ieeexplore.ieee.org/document/8694781

7 ANNEXES

1. TrainDataGenerator

```
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import Dropout
from keras import backend as K

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest',
    validation_split=0.2)
```

2. Generators

```
# Creer le generateur pour l'entrainement
train_generator = train_datagen.flow_from_directory(
    "../data_split/train",
    target_size=(255, 255),
    batch_size=32,
    class_mode='categorical',
    subset='training')

# Creer le generateur pour la validation
validation_generator = train_datagen.flow_from_directory(
    "../data_split/train",
    target_size=(255, 255),
    batch_size=32,
    class_mode='categorical',
    subset='validation')


 }
```