



## UNIVERSITÉ LYON 1

RAPPORT DE STAGE  
ELISE TECHNOLOGIES

---

# Attribuer un score de beauté à une image

---

*Elève :*

Rhizlaine DEGNI

*Tuteur :*

Eric DUPRE

17 septembre 2019



## Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Présentation de l'entreprise . . . . .	2
1.2	La Mission . . . . .	3
<b>2</b>	<b>Création du dataset</b>	<b>4</b>
2.1	l'algorithme ELO . . . . .	5
2.2	La création d'un bot Slack . . . . .	6
<b>3</b>	<b>Création d'un CNN</b>	<b>8</b>
3.1	Qu'est ce qu'un réseau de neurones? . . . . .	8
3.2	Comprendre ce qu'est un CNN . . . . .	9
3.3	CNN : Déetecter un beau selfie . . . . .	12
3.4	CNN : Déetecter une belle image . . . . .	16
<b>4</b>	<b>Conclusion</b>	<b>18</b>



# 1 Introduction

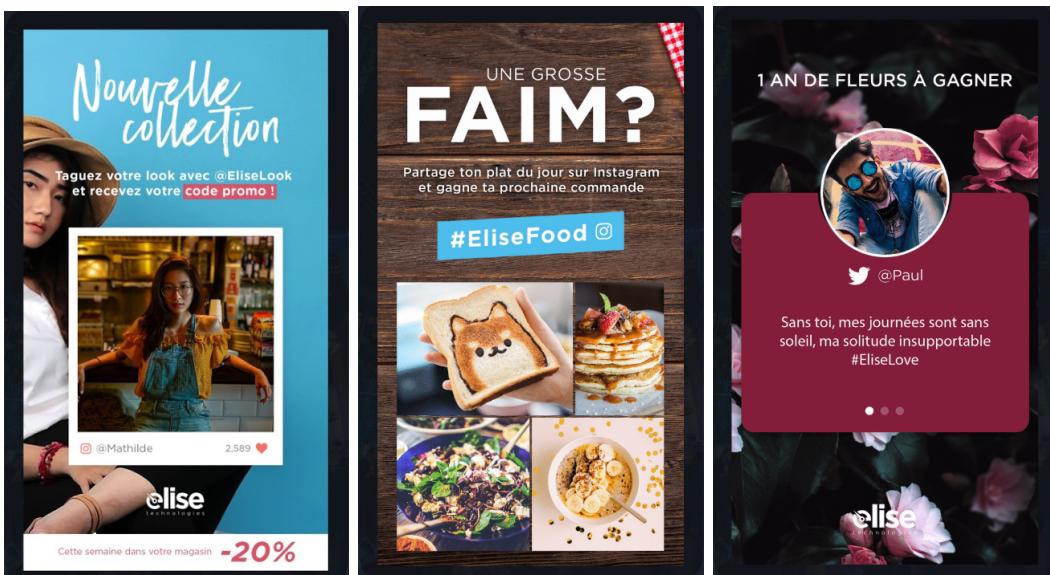
Dans le cadre de cet UE, j'ai pu réaliser un stage au sein d'Elise technologies pour une période de 6 mois. J'avais pour mission de trouver un algorithme de scoring d'images afin de faciliter le processus de modération des images. Pour comprendre davantage, il est nécessaire d'en savoir plus sur l'entreprise et son fonctionnement.

## 1.1 Présentation de l'entreprise

Élise technologies est une jeune start-up créée par Quentin Léchemia en 2014, d'abord sous le nom de My Band market. Élise pour Evolutionary Leading Intelligence Scoring Experiment. Elise Technologies est spécialisée dans l'analyse des interactions provenant des réseaux sociaux. Lors de sa création Élise fut d'abord concernée par l'industrie musicale. Elle permettait, à l'aide d'algorithmes, de détecter le prochain "buzz musical" grâce aux réseaux sociaux. Rapidement, l'algorithme est utilisé à d'autre fins. En 2016, Élise sort un social wall intelligent maximisant le nombre d'interactions générées par une audience. Et finalement, c'est en constatant certains manques de la publicité extérieure, tant sur le contenu que sur le tracking, que l'idée d'améliorer ce format pour les annonceurs a émergé.

À l'ère de la publicité digitale, il devient complexe pour les annonceurs de ne pas disposer d'une mesure précise de l'impact (branding et conversion) de leurs campagnes d'affichage. ELISE décide donc de proposer son format social et interactif pour le DOOH, un marché de 38 milliards d'euros en déployant une offre unique aux côtés des régies d'affichage et agences média.

Qu'est-ce que le DOOH ? Le DOOH est le digital out of home. Il s'agit de la publicitaire extérieure digitale. Il s'agit dans d'autres mots des publicités situés sur les panneaux publicitaires à écran. Élise propose donc essentiellement à ce jour, une solution permettant de créer des publicités extérieures digitales avec du contenu social directement récupérer sur divers réseaux sociaux tels que Facebook, Twitter et Instagram.



L'utilisation des réseaux sociaux a de nombreux avantages, notamment marketing. En effet, cela permet de faire participer directement les gens à la publicité. Cela permet alors dans un premier temps à rendre la publicité beaucoup plus dynamique et plus interpellante pour les passants. La publicité se fait directement par de potentiels clients. Cela permet d'être plus proche du passant et de créer un plus grands sentiments de confiance. En effet, l'avis d'autres consommateurs paraît plus "honnête" que celui de l'annonceur. Dans un deuxième temps cela permet d'étendre l'audience. Une personne passant devant le panneau publicitaire qui participe à la publicité permet également de diffuser la publicité dans son réseaux (ses amis Facebook, ses followers sur instagram ou twitter, etc.).

De plus, cela permet de répondre une grande problématique de la publicité extérieure. Effectivement, sans mettre de caméra devant les écrans pour compter le nombre de personnes intéressé par la publicité, il est impossible de mesurer réellement la portée d'une opération publicitaire. En faisant participer les personnes sur les réseaux sociaux, nous pouvons plus facilement mesurer cet impact, grâce aux nombres de participations, de "likes" générés ou bien même des réactions commentées. On peut également avoir plus d'informations sur les catégories de personnes impactées par la publicité (par exemple, plus de réactions chez les femmes de moins de 45 ans). Cependant, ce dernier point est rendu plus difficile avec la loi sur la RGPD qui empêche de récupérer les informations personnelles d'une personne sans son accord.

Ce processus a aussi des désavantages, le plus important étant qu'elle génère une baisse de contrôle sur la publicité. En effet, contrairement à une campagne de publicité ordinaire, le contenu est variable et est entre les mains du grand public. Il est donc nécessaire de contrôler ce qui sera affiché sur les écrans grâce à une modération des postes sociaux affichés. Pour chaque opération, nous avons donc besoin d'un modérateur qui contrôle toutes les participations générées. Il paraît impensable d'afficher un poste peut avantager pour l'annonceur par exemple, ou même un poste ne correspondant pas aux attentes. Il faut aussi veiller à ce que les postes respectent les lois régissant le monde de la publicité. On ne peut accepter des postes présentant de la nudité, ou bien du tabac, de l'alcool ou des mineures. La modération prend donc beaucoup de temps.

Il existe différentes possibilités de postes sociaux. On peut retrouver du texte, des photos ou des vidéos. Pour faciliter la modération, Élise utilise différentes méthodes pour pré-filtrer les images. Cependant, ces méthodes ne concernent que le texte. En effet, l'utilisation de mot clés à bannir permet de faire un premier tri des messages. Pour les images, cela est beaucoup plus complexe. C'est dans cette idée de trouver un moyen de faciliter la modération des images, que la mission de mon stage est née.

## 1.2 La Mission

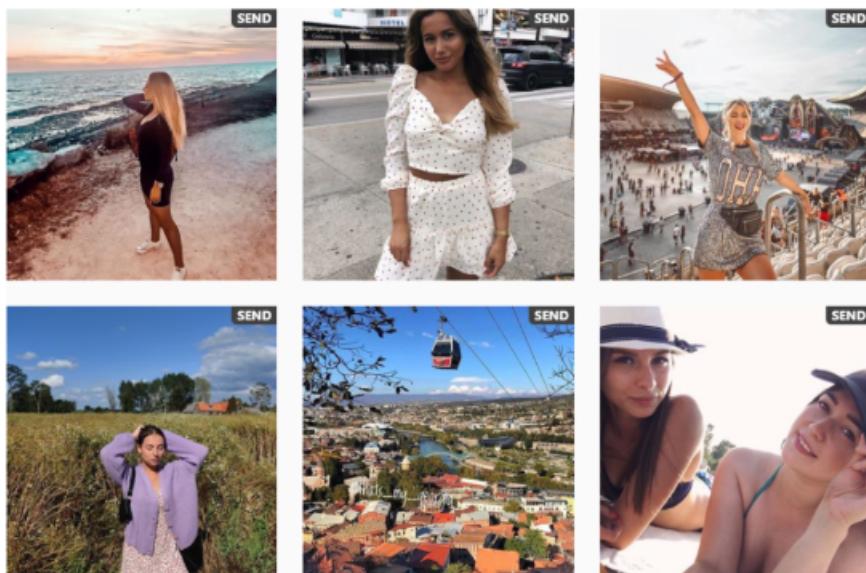
Il m'a été demandé lors de ce stage, de trouver une solution qui pourrait permettre de faciliter la modération des images. Nous avons décidé alors de trouver un moyen de privilier les belles images. En effet, nous cherchons à trouver les images les plus "DOOHable", c'est-à-dire celle qui serait le plus susceptible d'apparaître sur une publicité extérieure. On évite ainsi, les images de mauvaise qualité, mal cadrés, avec peu de piqué etc. Pour répondre à cette problématique, nous avons tout de suite pensé à un réseau de neurones convolutionnel capables de détecter ses différents critères qui détectent une belle image susceptible d'être présentée sur les panneaux de publicités. Nous verrons en détail plus tard le choix d'un réseaux de neurones.



Pour pouvoir entraîner ce modèle, nous avions besoin d'un dataset d'image et une mesure de leur beauté répondant au critère de l'entreprise, de telle sorte que le modèle puisse "penser" comme un membre de l'entreprise pour la modération. C'est pour cela qu'il était nécessaire d'avoir un dataset propre à l'entreprise. La première tâche de ma mission a donc été la création d'un dataset. Pour la réalisation de ma mission, j'avais à disposition tous les services disponibles sur Amazon AWS, ce qui m'a permis d'avoir tous les outils nécessaires à la réalisation de cette mission.

## 2 Crédit du dataset

La création d'un dataset, s'est effectué en deux étapes. La première consiste à récupérer des images des réseaux sociaux afin de garnir notre dataset. Pour cela, il a fallu trouver un moyen de récupérer des images directement depuis un réseau social et de les stocker sur un S3. Un S3 est un service d'hébergement proposé par Amazon AWS. Pour cela, j'ai créé une extension chrome à l'aide de TemperMonkey qui détecte les images présentent sur une page web sur Instagram ou Twitter. Sur chaque image un bouton était ajouté permettant de télécharger l'image directement sur le S3.



La partie la plus complexe dans la création du dataset a été la labélisation. En effet, il a fallu trouver tout un système de labélisation des images efficace et non biaisé pour cela nous avons pensé à faire voter les images par les membres de l'entreprise qui connaisse les critères de beauté attendue. Afin de limiter les biais des votes, nous avons choisi de faire un vote entre deux images plutôt que de noter une image, car il est plus évident de comparer deux images plutôt que d'en juger une. Pour pouvoir classer les images, grâce aux votes deux à deux des images, nous avons utilisé l'algorithme ELO [3]. Il s'agit de l'algorithme de classement utilisé historiquement lors des concours d'échec.



## 2.1 l'algorithme ELO

Cet algorithme est historiquement utilisé pour évaluer le niveau de capacités relatives d'un joueur d'échecs. Il est aussi utilisé dans les jeux vidéo afin de mesurer le niveau relatif d'un joueur. Il a d'ailleurs été utilisé par Mark Zuckerberg pour établir le classement des filles d'Harvard. Le classement ELO est un classement par point. Nous attribuons un score de départ à 0 à chacune des images. Ce score va évoluer selon les résultats des "matchs" des images, c'est-à-dire qu'en une image doit être comparé à une autre. Le score d'une image est donc augmenté lorsqu'elle remporte le match et diminué en cas de défaite. L'augmentation ou la diminution du score ne sera pas la même selon la difficulté du match car il va dépendre du score de l'image adverse. Pour cela, nous calculons la probabilité d'une image I1 de gagner face à une autre image I2.

$$\text{estimation}_{I1} = \frac{1}{1 + 10^{\frac{ELO_{I2} - ELO_{I1}}{200}}} \quad (1)$$

Par exemple avec une image I1 qui a un score de 25 et une autre image I2 avec un score de 12. Si la personne préfère l'image I1 à l'image I2, les scores seront modifiés ainsi.

$$\text{score}_{I1} = 25 + 10(1 - \text{estimation}_{I1}) = 29.62 \quad (2)$$

$$\text{score}_{I2} = 12 + 10(0 - \text{estimation}_{I2}) = 7.37 \quad (3)$$

Si en revanche l'image I2 gagne on aurait eu :

$$\text{score}_{I1} = 25 + 10(0 - \text{estimation}_{I1}) = 19.62 \quad (4)$$

$$\text{score}_{I2} = 12 + 10(1 - \text{estimation}_{I2}) = 17.37 \quad (5)$$

Avec cet algorithme, au bout d'un certain nombre de votes le classement des scores se stabilise. Cet algorithme s'adapte bien aux erreurs de jugement. En effet si une personne d'aimer pas une image généralement aimée, l'image sera moins pénalisée car cela ne correspondra pas au classement déjà établi et donc diminuera beaucoup pour rester proche de sa place. Afin d'estimer, le nombre de votes nécessaires pour arriver à une stabilité des scores, une étude plus poussée à dû être mis en place. Nous avons donc réalisé plusieurs simulations. Nous avons simulé les erreurs de jugement et mesuré le nombre de votes nécessaires pour différentes quantités d'images afin de définir la taille du dataset pour que la labélisation puisse se faire dans un temps raisonnable. Nous avons suite au résultat ci-dessous, choisi un dataset de 1500 images.

Nb d'images	Nb de votes pour 75% précision
1000	20000
2000	40000
3000	60000
4000	80000
5000	100000



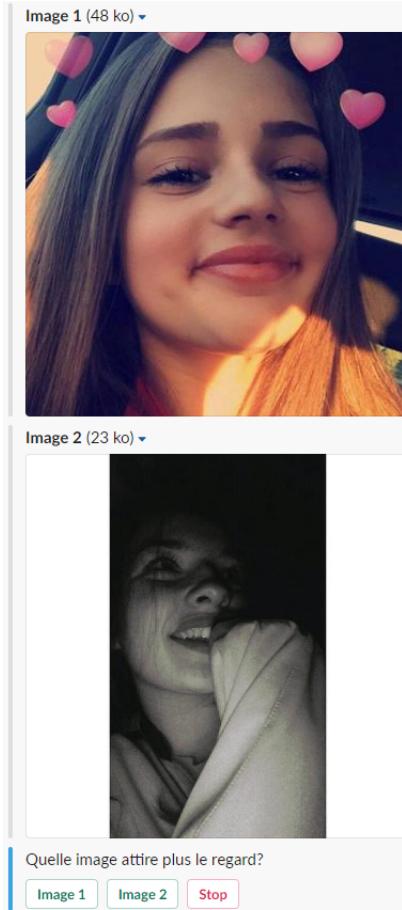
## 2.2 La création d'un bot Slack

Le nombre de votes nécessaires étant important, il a fallu mettre en place un dispositif accessible à toute l'entreprise afin de pouvoir voter facilement depuis leur ordinateur ou leur téléphone. Nous avons donc créé un bot slack qui propose un vote entre deux images. Pour savoir quelles images proposées nous avons dû faire plusieurs simulations. Nous avons pu alors remarquer qu'il était plus pertinent de proposer dans un premier temps un vote aléatoire avec une probabilité de tirages plus basses pour les images déjà tirées plusieurs fois de façon à ce que toutes les images puissent faire un certain nombre de matchs afin d'avoir une première idée de leur classement.

Afin d'accélérer la convergence vers un état stable, faire voter des images similaires permet d'ajuster plus rapidement les images. En effet, au bout d'un certain nombre de votes, il est inutile d'affronter une image avec un score faible et une image avec un score élevé, car nous sommes quasiment sur de l'issue du vote. C'est pour cela, qu'on obtient de meilleur résultat lorsqu'on incorpore des votes par scores similaires. Voici les résultats des simulations. On peut voir ci-dessous le pourcentage de votes nécessaires en aléatoire. On voit très clairement que les résultats sont meilleurs lorsque les 75% premiers matchs sont générés de manières aléatoires puis par scores proches. Cela permet donc de converger plus rapidement vers un état stable.

Nb d'images	Nb de votes	Précision avec 100% d'aléatoire	75%	50%	25%
1000	20000	0.745519	0.784162	0.751630	0.644485
2000	40000	0.751015	0.801088	0.738721	0.662126
3000	60000	0.753457	0.788264	0.739679	0.658771
4000	80000	0.750251	0.7843430	0.736300	0.658590
5000	100000	0.748543	0.779992	0.737368	0.650132

Le bot slack propose donc aléatoirement deux images puis des images proches en matière de score. Les scores sont conservés dans une base de données sur AWS, une dynamo Db, et y sont modifiés à chaque match. On peut ainsi labelliser notre dataset. On aura un classement d'images par score. On pourra alors déterminer différents seuils séparant les belles images des moins belles. Voici en image le Bot créé.



Nous avons fait le choix d'utiliser uniquement des images de selfie. Il y a deux raisons à ce choix, la première et étant que nous avons opté pour un dataset d'uniquement 1500 images afin de pouvoir les labelliser. On se restreint donc à une catégorie d'image. Cela permet de créer un modèle plus performant sur les selfies souvent utilisé sur les réseaux sociaux. La deuxième raison est qu'il est de limiter le biais lors des votes. En effet, on préfère peut-être plus une image d'une personne à une image de paysage par exemple. Il est plus facile d'évaluer deux images d'une même catégorie. Cela évite de voter plus pour une catégorie qu'une autre.

On a donc, avec tout cela, pu récupérer 1500 selfies sur Instagram et Twitter pour alimenter notre bot et permettre la labélisation de ces images. Cependant cette labélisation nécessite la participation de tous les membres de l'entreprise et du temps, et n'étant pas une priorité nous avons décidé de trouver un modèle de scoring d'image sur d'autres datasets déjà existants mais ne répondant pas spécialement aux critères de beauté attendue par l'entreprise. L'essentiel était d'avoir à disposition un modèle capable de repérer des critères de beauté et la possibilité de l'entraîner sur des critères propres à l'entreprise.

Nous avons donc utilisé un dataset de selfies labelisé selon des votes obtenus que c'est images, des votes venant d'internautes. Mais avant de ce lancer dans cette tâche. Il a fallu un temps de recherche consacré à la compréhension des réseaux de neurones convolutionnel, très efficace sur la détection de features sur les images. Il est donc nécessaire d'avoir quelques explications sur ce qu'est un CNN (convolutional neural network).



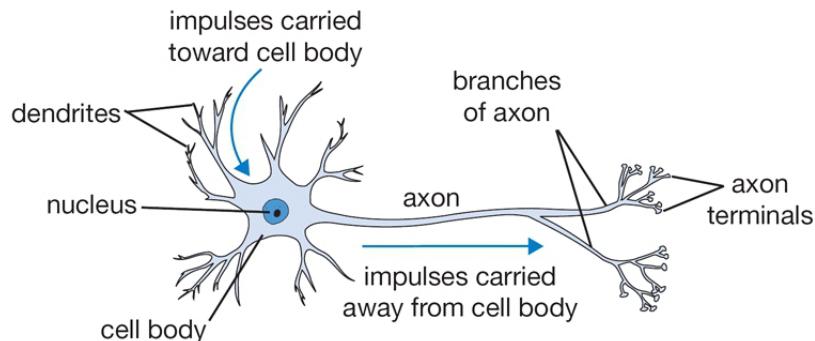
## 3 Crédit d'un CNN

Un réseau de neurones convolutionnel est un type de réseaux de neurones souvent utilisé pour la classification d'image. Pour comprendre ce qu'est un CNN, il est nécessaire de comprendre le fonctionnement d'un réseau de neurones. [2][4][1]

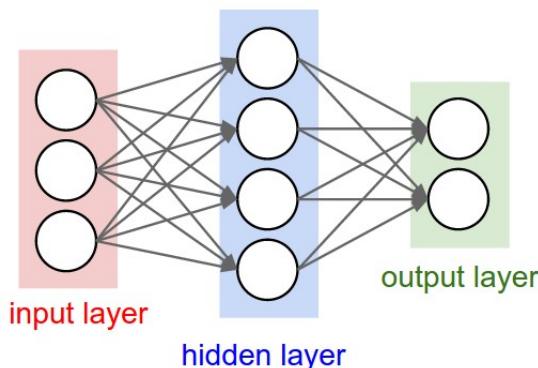
### 3.1 Qu'est ce qu'un réseau de neurones ?

Les réseaux de neurones sont initialement inspirés par les systèmes neuronaux biologiques, et permettent aujourd'hui l'obtention de bons résultats dans les tâches d'apprentissage automatique.

En effet, dans un système neuronal biologique, chaque neurone reçoit des signaux d'entrée de ses dendrites et produit des signaux de sortie le long de son axone. L'axone finit par se ramifier et se connecter via les synapses aux dendrites d'autres neurones. Dans le modèle de calcul d'un neurone, les signaux qui voyagent le long des axones interagissent de manière multiplicative avec les dendrites de l'autre neurone en fonction de la force synaptique de cette synapse. Cette force peut être positive ou négative. Dans le modèle de base, les dendrites transmettent le signal au corps de la cellule où ils sont tous additionnés. Si la somme finale dépasse un certain seuil, le neurone peut déclencher, envoyant un signal le long de son axone. Sur la base de cette interprétation, nous modélisons le processus de déclenchement du neurone avec une fonction d'activation, qui représente la fréquence des signaux le long de l'axone.



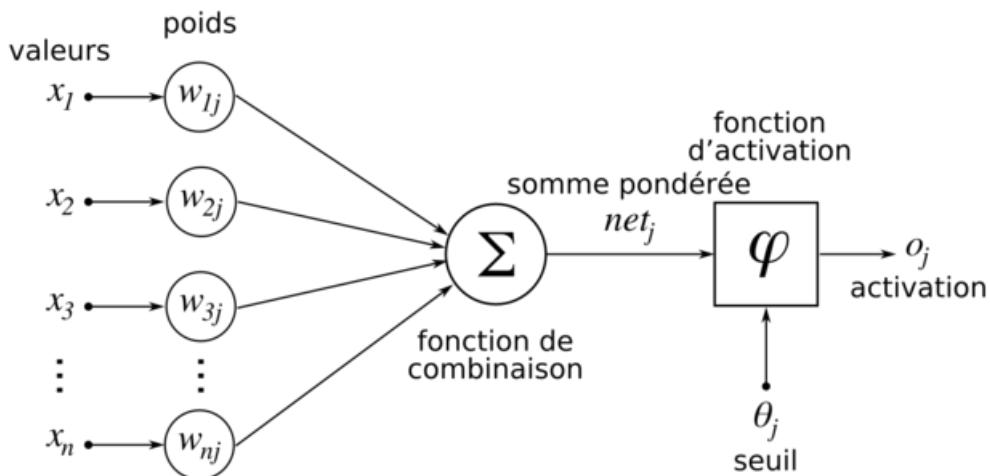
Les réseaux de neurones sont modélisés comme des ensembles de neurones connectés dans un graphe acyclique. En d'autres termes, les sorties de certains neurones peuvent devenir des entrées pour d'autres neurones. Pour les réseaux de neurones normaux, le type de couche le plus courant est la couche entièrement connectée comme ci-dessous.





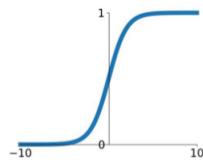
Chaque neurone de la couche cachée va calculer la somme de ses entrées puis passer cette valeur à travers la fonction d'activation pour produire sa sortie, de la manière suivante.

$$o_j = \varphi\left(\sum_{i=1}^n x_i * w_i - \theta_j\right) \quad (6)$$

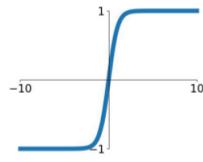


Il existe différentes fonctions d'activation. Voici les plus utilisées :

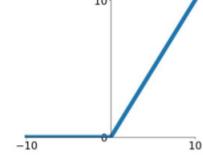
**Sigmoid**  
 $\sigma(x) = \frac{1}{1+e^{-x}}$



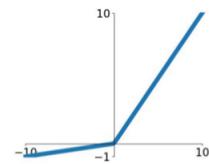
**tanh**  
 $\tanh(x)$



**ReLU**  
 $\max(0, x)$



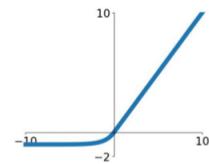
**Leaky ReLU**  
 $\max(0.1x, x)$



**Maxout**  
 $\max(w_1^T x + b_1, w_2^T x + b_2)$

**ELU**  

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

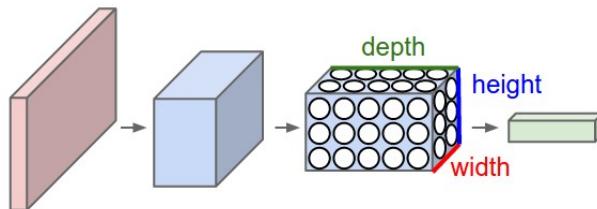


### 3.2 Comprendre ce qu'est un CNN

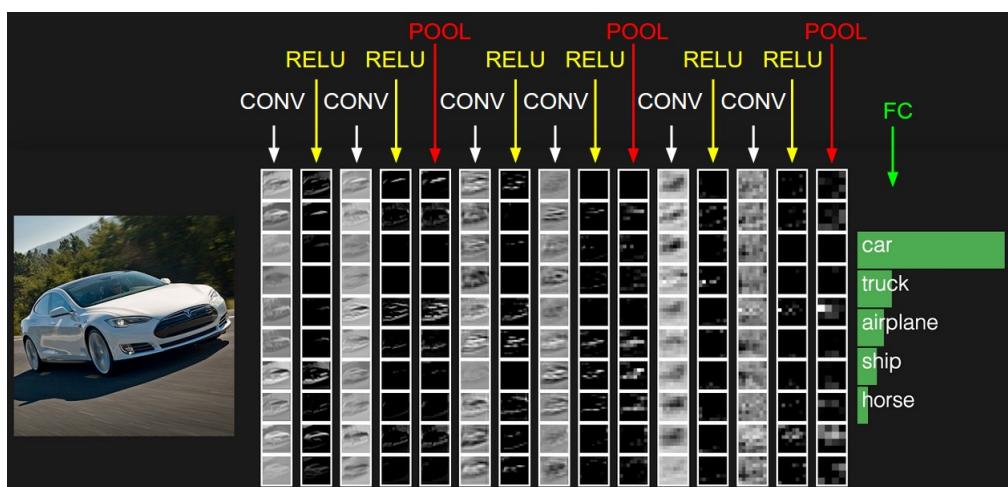
Les réseaux de neurones convolutionnels sont constitués de neurones dont les poids et les biais peuvent être appris. Chaque neurone reçoit des entrées et effectue un produit scalaire. L'ensemble du réseau exprime toujours une seule fonction de score différentiable : des pixels d'image d'un côté aux scores de classe de l'autre. Et ils ont toujours une fonction de coût sur la dernière couche.



Les architectures ConvNet supposent explicitement que les entrées sont des images, ce qui nous permet de coder certaines propriétés dans l'architecture. Celles-ci rendent alors la fonction d'activation plus efficace à mettre en œuvre et réduisent considérablement la quantité de paramètres dans le modèle.

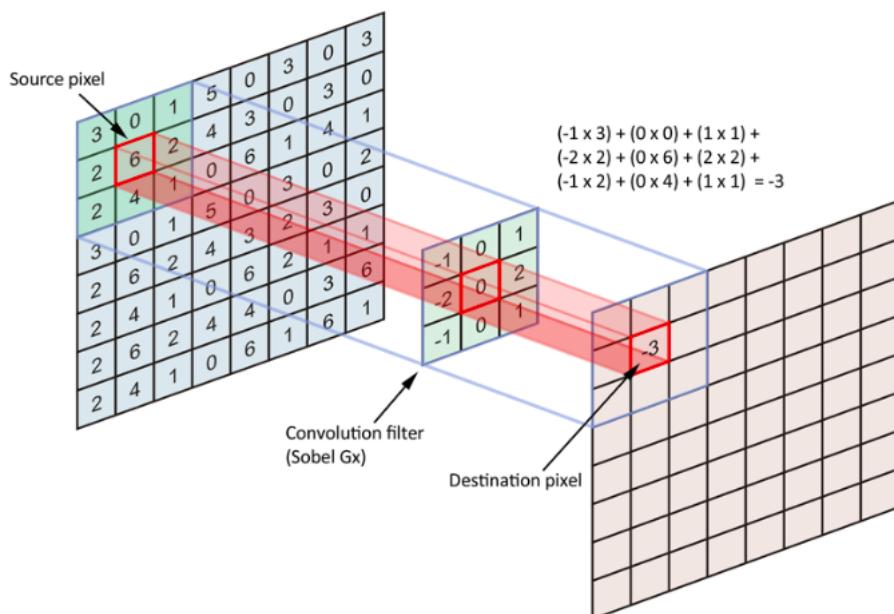


Un ConvNet simple est une séquence de couches et chaque couche d'un ConvNet transforme un volume d'activation en un autre à l'aide d'une fonction différentiable. Nous utilisons trois types de couches pour construire des architectures ConvNet : couche convolutif, couche Pooling et couche entièrement connectée. Nous allons empiler ces couches pour former une architecture ConvNet complète.

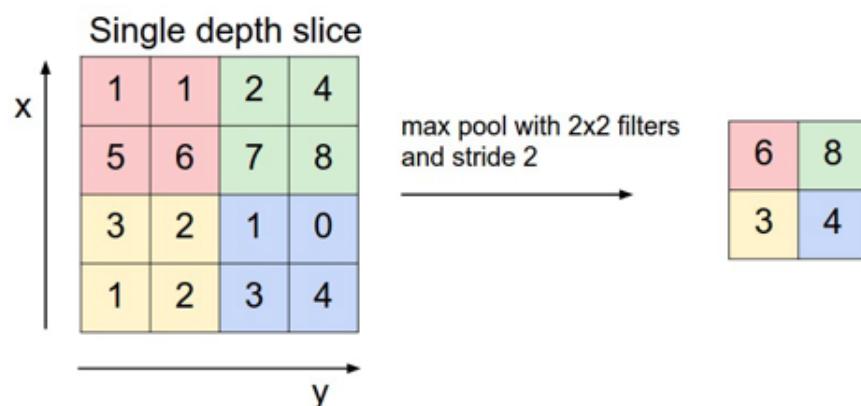




La couche de convolution est la composante clé des réseaux de neurones convolutifs, et constitue toujours au moins leur première couche. Cette couche consiste à faire glisser un filtre sur la largeur et la hauteur du volume d'entrée, ainsi nous produisons une carte d'activation qui donne les réponses de ce filtre à chaque position spatiale. Intuitivement, le réseau apprendra les filtres qui s'activent lorsqu'il détecte un type de caractéristique visuelle, tel qu'un bord d'une orientation ou une tache de couleur sur la première couche, ou éventuellement des motifs complets.

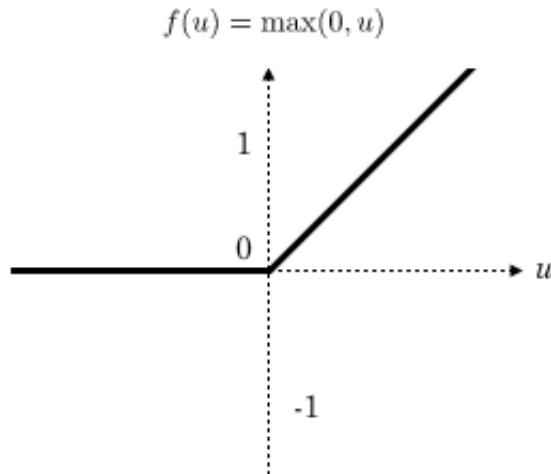


On obtient pour chaque paire (image, filtre) une carte d'activation, ou feature map, qui nous indique où se situent les features dans l'image : plus la valeur est élevée, plus l'endroit correspondant dans l'image ressemble à la feature. Les noyaux des filtres sont les poids de la couche de convolution. Ils sont initialisés puis mis à jour par rétropropagation du gradient. Une couche Pooling est souvent placée entre deux couches de convolution : elle reçoit en entrée plusieurs features map, et appliquent à chacune d'entre elles l'opération de pooling. Elle consiste à réduire la taille des images, tout en préservant leurs caractéristiques importantes. On améliore ainsi l'efficacité du réseau et on évite le sur-apprentissage. Il en existe deux principaux, la première consiste à réduire une zone en faisant la moyenne, l'autre en récupérant la valeur maximale.





Dans l'exemple page 9 de la voiture, on retrouve aussi des couches de correction, ReLU.



La couche de correction ReLU remplace donc toutes les valeurs négatives reçues en entrées par des zéros. Elle joue le rôle de fonction d'activation.

La couche fully-connected (FC) constitue toujours la dernière couche d'un réseau de neurones, convolutif ou non.

Ce type de couche reçoit un vecteur en entrée et produit un nouveau vecteur en sortie. Pour cela, elle applique une combinaison linéaire puis éventuellement une fonction d'activation aux valeurs reçues en entrée.

La dernière couche fully-connected permet de classifier l'image en entrée du réseau : elle renvoie un vecteur de taille N, où N est le nombre de classes dans notre problème de classification d'images. Chaque élément du vecteur indique la probabilité pour l'image en entrée d'appartenir à une classe.

La couche fully-connected détermine le lien entre la position des features dans l'image et une classe. En effet, le tableau en entrée étant le résultat de la couche précédente, il correspond à une carte d'activation pour une feature donnée : les valeurs élevées indiquent la localisation (plus ou moins précise selon le pooling) de cette feature dans l'image. Si la localisation d'une feature à un certain endroit de l'image est caractéristique d'une certaine classe, alors on accorde un poids important à la valeur correspondante dans le tableau. On retrouve donc en sortie de notre modèle la probabilité d'appartenance à l'image pour chacune des classes.

### 3.3 CNN : Déetecter un beau selfie

Afin de mettre en pratique ses connaissances, j'ai pu lors de mon stage commencer une première implementation d'un CNN pour un problème classique, celui de la reconnaissance de numéros à l'aide du dataset MNIST. Dans un deuxième temps, nous avons cherché à répondre à notre problématique : Est-il possible de différencier les belles images des mauvaises. Comme nous avons vu précédemment, nous nous sommes dans un premier temps concentré sur des images de selfies. Notre dataset d'état pas encore labelisé, il a fallu trouver un dataset déjà existant regroupant des selfies.



Nous avons donc choisi le dataset SCUT [6], regroupant 5500 selfies d'hommes et de femmes. Toutes les images ont été labellisées avec un score de beauté comprise entre 1 et 5 par 60 volontaires âgés de 18 à 27 ans, plus le score est élevé plus la photo est belle.

Pour ce faire, nous avons choisi la méthode de transfert learning. Le transfert learning consiste à utiliser un modèle déjà entraîné sur des problématiques plus ou moins similaires et d'ajuster ses poids sur notre problématique. Nous avons choisi dans un premier temps à tester le modèle Densenet201 car ce modèle a été utilisé à plusieurs reprises lors d'un challenge kaggle dont le but est d'identifier le smartphone avec lequel des images avaient été prises. On peut donc penser que ce modèle arrive à repérer les différentes qualités d'image et pourrait alors servir à notre problème. En effet la qualité de l'image joue dans l'évaluation de sa beauté.

Dans un CNN les images d'entrées doivent toutes avoir le même format, il est donc nécessaire de réaliser un traitement des images pour pouvoir avoir le format d'image. Ici, notre dataset contient déjà des images de tailles similaires. Il nous reste donc plus à réentraîner notre modèle Densenet201 sur notre dataset, c'est-à-dire calculer les différents poids de notre modèle pour chaque photo pour de manière à limiter les erreurs entre les sorties de notre modèle et la valeur réelle du score de l'image. Pour cela nous entraînons notre modèle sur une partie de nos données et évaluons la qualité du modèle en regardant ses erreurs sur les autres données.

Nous entraînons notre modèle sur plusieurs époques, à chacune de ses époques, le modèle recalcule les poids en fonction des erreurs précédentes. Cette erreur est calculée, grâce à la fonction de cout, nous avons utilisé ici la RMSE (Root Mean square Error). Nous arrêtons l'entraînement, lorsque l'erreur ne diminue plus assez. Au bout de 10 époques, notre modèle stagne. Nous pouvions donc évaluer la qualité de notre modèle finale. Pour l'évaluer, nous avons regardé 3 critères la RMSE, la MAE et la corrélation de Pearson.

La RMSE est l'erreur quadratique moyenne. Elle représente l'erreur moyenne commise entre le score prédict d'une image par notre modèle et son score réelle. Elle est calculée comme ci-dessous :

$$RMSE = \sqrt{\sum_{i=1}^N \frac{(Y_{pred_i} - Y_{true_i})^2}{N}} \quad (7)$$

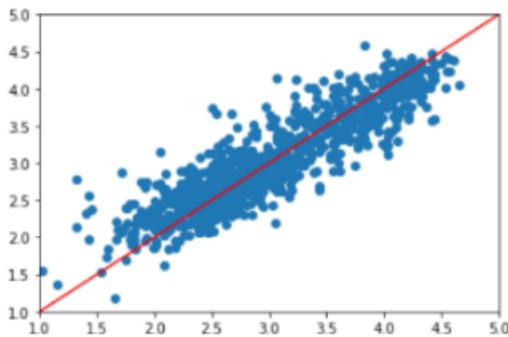
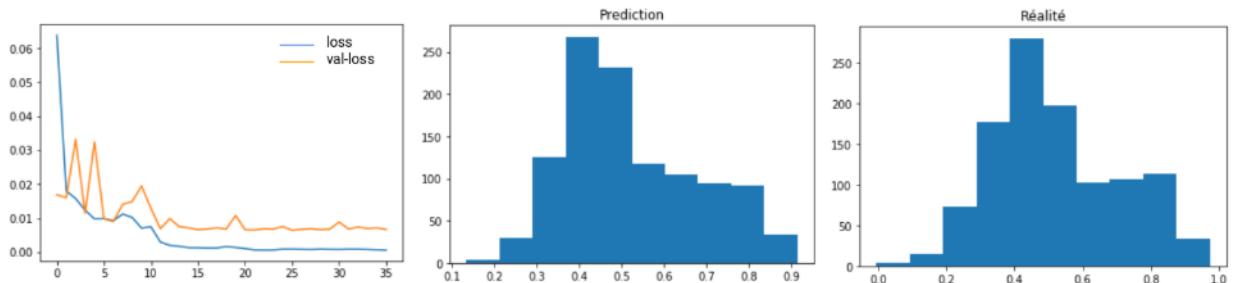
La MAE est l'erreur absolue moyenne.

$$MAE = \sum_{i=1}^N \frac{|Y_{pred_i} - Y_{true_i}|}{N} \quad (8)$$

Le coefficient de corrélation de Pearson (PC), est un indicateur de corrélation entre les scores prédicts et les scores réels. Cela permet d'avoir une bonne idée de la qualité du modèle. Plus il est proche de 1 plus les chances que notre modèle est compris la réalité sont grandes.

$$PC = \frac{N \sum Y_{pred} * Y_{true} - (\sum Y_{pred} \sum Y_{true})}{\sqrt{[N \sum Y_{pred}^2 - (\sum Y_{pred})^2][N \sum Y_{true}^2 - (\sum Y_{true})^2]}} \quad (9)$$

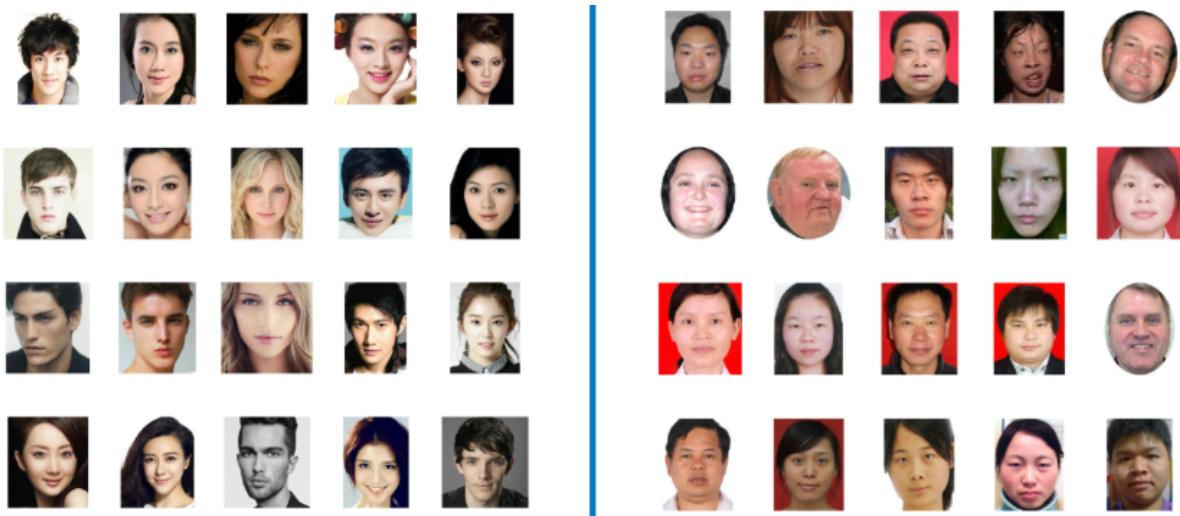
Pour optimiser, notre modèle nous avons normalisé les scores, pour de meilleurs résultats, c'est-à-dire qu'on les ramène dans un intervalle compris entre 0 et 1. Nous avons obtenu les résultats suivants.



Nombre d'époques: 20 (early stopping)  
Last checkpoint: 10

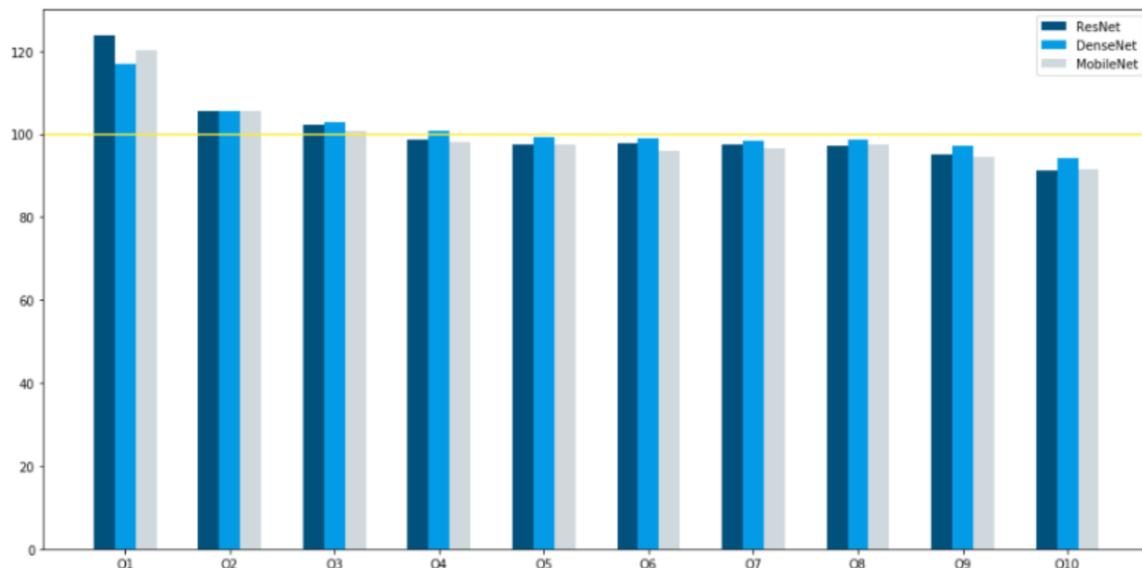
<b>RMSE</b>	0.080	<b>RMSE inverted</b>	0.30
<b>MAE</b>	0.060	<b>MAE inverted</b>	0.224
<b>PC</b>	0.90	<b>PC inverted</b>	0.90

Les 4 graphiques représentent, dans l'ordre, l'évolution des erreurs sur les données d'entraînement et les données test au cours des époques, la répartition des scores prédicts, la répartition des scores réels, un nuage de points des scores réels et des scores prédicts pour chaque image. On peut donc voir sur c'est graphique une similitude entre les scores prédicts et les scores réels. Nous pouvons voir dans ce résumé de l'évaluation de notre modèle que l'erreur moyenne sur nos scores normalisés est de 0.08, ce qui fait une erreur moyenne de 0.30 sur les scores compris entre 0 et 5. Le modèle s'est entraîné pendant 20 époques et la 10ième époques est celle où le modèle a fait le moins d'erreurs. Notre coefficient de Pearson est de 0.9, ce qui est proche de 1, comme souhaité. Nous pouvons également vérifier le modèle visuellement.

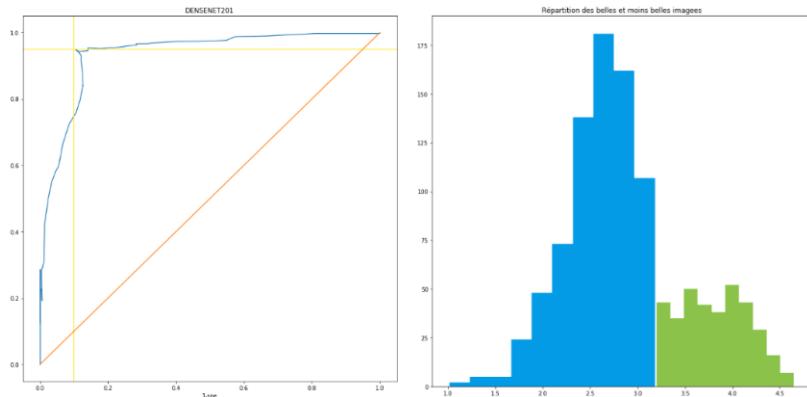


On peut voir à gauche les images avec score prédict élevé et à droite, un score prédict faible. On observe que le modèle attribue des scores élevés aux beaux selfies et un score faible au moins beaux selfies. Le modèle répond bien à la problématique.

Nous avons aussi essayé d'ajouter d'autres couches à notre modèle pour le rendre plus performant, mais les résultats étaient moins bons. Nous avons également testé le transfert learning avec d'autres modèles déjà entraînés comme le resnet ou bien le mobilenet, des modèles utilisés pour la détection d'objets sur des images. Ces deux modèles ont donné des résultats satisfaisants mais légèrement inférieurs à ceux obtenus grâce aux Densenet201. La différence étant minime, nous avons décidé de comparer ces trois modèles en détails. En effet, nous souhaitons un modèle bon parmi les scores très élevés et très faibles. Nous voulons être sûre à détecter les plus belles images et de ne pas prendre les images les moins belles. Il est donc important de regarder les erreurs commises pour différents niveaux de score. Nous avons donc découpé l'intervalle de score en 10 et nous avons regardé les erreurs sur les 10 parties.



On peut voir ici, que notre modèle Densenet201 est plus efficace que les autres. Nous avons alors cherché à savoir s'il existait un seuil optimal permettant de différencier les belles images des moins belles.



Pour cela on observe pour différents seuils, la sensibilité et la spécificité de notre modèle. La sensibilité est la part de belles images prédites sur toutes les belles images réelles. La spécificité est des moins belles images prédites sur toutes les moins belles images. Ce qui nous permet de tracer la courbe ROC et de déterminer le seuil optimal, c'est-à-dire le score qui permet de mieux séparer les belles images des moins belles. On remarque que ce seuil coupe nos données telles que les 2/3 des données sont des moins belles images et le 1/3 sont de belles images. A ce seuil 89% des images que nous prédisons comme belles le sont et 95% des images que nous prédisons comme moins belles le sont.

Nous nous sommes également assuré que le modèle soit robuste. C'est-à-dire que les erreurs ne dépendent pas des données qui ont servi à l'entrainer. Pour cela, nous avons observé nos mesures de qualité du modèle pour différentes données d'entrainement grâce à une cross-validation.

Fold	1	2	3	4	Average
RMSE	0.080	0.080	0.073	0.05	0.07
MAE	0.060	0.061	0.057	0.039	0.054
PC	0.90	0.908	0.927	0.966	0.925

On peut donc voir que notre modèle est robuste. Cependant, notre modèle est-il toujours efficace sur des images qui ne sont pas des types selfies ?

### 3.4 CNN : Déetecter une belle image

Nous allons donc maintenant chercher à prédire une belle image que ce soit un selfie, un paysage, une photo d'un animal etc. Pour cela nous avons besoin d'un autre dataset. Nous avons utilisé le dataset AVA. Il contient plus de 255 000 images notées sur leur beauté par des photographes amateurs. Chaque photo a été voté par en moyenne 200 personnes lors de concours de photographie. Les notes vont de 1 à 10, avec 10 le score de beauté le plus élevé. Nous nous sommes pour cela intéressés aux travaux de recherche réalisés par Hossein Talebi and Peyman Milanfar appelés NIMA [5].

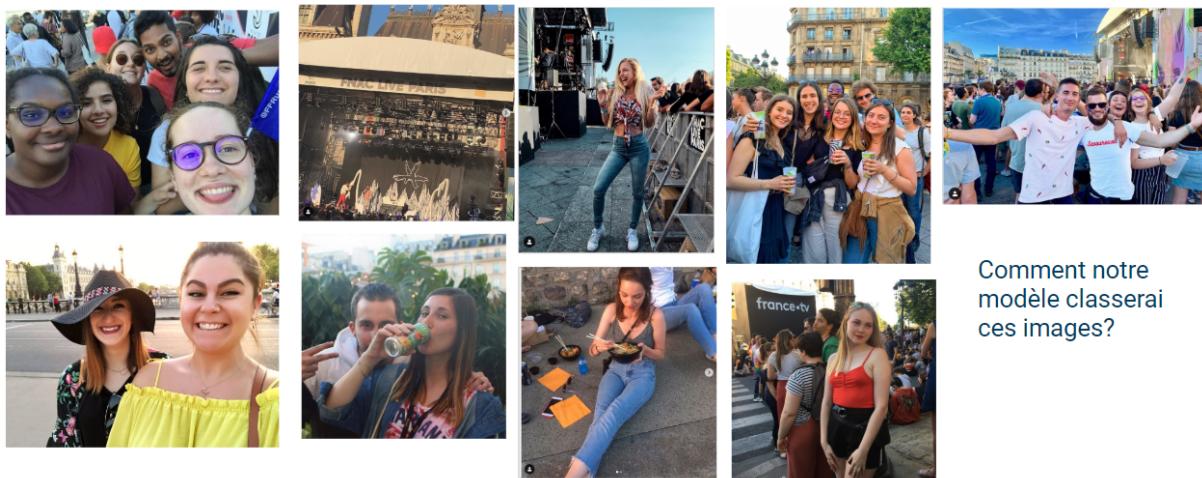


Sur le même principe que le CNN réalisé pour les selfies. Nous utilisons ici des modèles déjà entraînés. Pour ce dataset, un traitement d'images est nécessaire afin d'avoir un format unique de celles-ci. Comme présenté dans leur recherche, les images sont redimensionnées en 256X256 puis on récupère aléatoirement une partie de l'image de taille 224X224. La fonction de coût utilisée est la EMD.

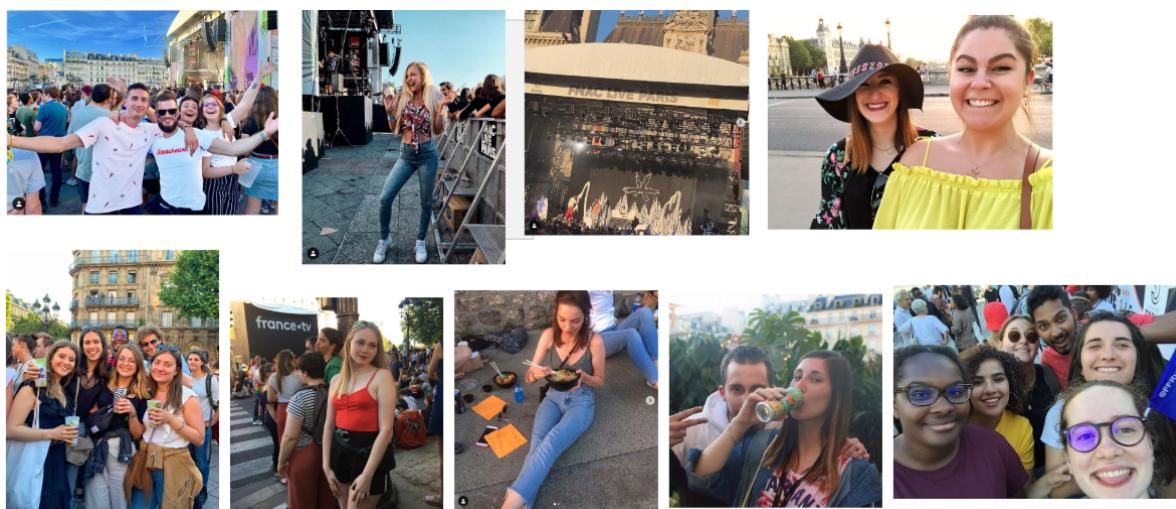
$$EMD = \sqrt{\frac{|SC_{Y_{true}} - SC_{Y_{pred}}|^2}{N}}$$

, SC = somme cumlative

Le modèle NIMA obtient les meilleurs résultats en réentraînant les modèles Mobilenet, NasNet et InceptionV2 avec des EMD allant de 0.08 à 0.06. Nous avons donc évalué si ce modèle permettait d'aider actuellement la modération de l'entreprise à accéder plus facilement aux plus belles images. C'est pourquoi nous avons testé de prédire le score de nouvelles images prises sur les réseaux sociaux et nous avons regardé si le classement par score de ses images était identique à l'ordre de préférence de l'entreprise.



Voici ce que l'on obtiendrait comme classement dans la modération, de gauche à droite et de haut en bas.





On obtient en priorité des images colorées et lumineuses. Un classement qui correspond majoritairement aux attentes de l'entreprise en critère de beauté. Cependant ces résultats peuvent être davantage fidèles aux idées de l'entreprise grâce à une labélisation des données propres à l'entreprise. D'où la possibilité d'utiliser en temps voulu, le dataset que nous avons créé, notamment grâce au bot slack. Afin de prévenir la croissance de l'entreprise et donc la nécessité d'une modération plus rapide, nous avons commencé à regarder comment intégrer ce modèle au pipeline de l'entreprise. J'ai donc pu découvrir Sagemaker, l'outil d'amazon permettant de créer, former et déployer des modèles. Sagemaker permet de sauvegarder le modèle directement sur un S3 d'AWS et de créer un endpoint vers le modèle. L'application Élise hébergée sur AWS peut donc interroger directement l'endpoint pour récupérer les scores des images collectés afin de les classer de la meilleure à la moins belle. Nous avons donc répondu à la mission, nous sommes capable de faciliter la modération en détectant les belles images afin d'accélérer leur modération. Il sera également possible de fixer un score de beauté pour lequel les images avec un score supérieur seront automatiquement validé en modération.

## 4 Conclusion

Ce stage a été une très belle expérience. Travailler dans une jeune start up m'a permis d'apprendre à être autonome, rigoureuse et à avoir confiance en mon travail. Lors de ce stage, j'ai appris à travailler avec les différents outils de AWS. J'ai pu créer des Lambdas, un service d'AWS qui permet d'exécuter du code. J'ai pu avec ses Lambdas créer mon bot slack. La création de mon bot m'a également permis de me familiariser avec les requêtes HTTP vers une API. J'ai également utilisé des EC2, un service Web d'AWS qui fournit une capacité de calcul sécurisée et redimensionnable dans le cloud. J'ai alors pu créer des environnements de travail distant spécialisé dans le deep-learning avec une surcouche tensorflow. J'ai construit également des bases de données grâce à dynamoDB, le service AWS pour la création de BD. J'ai pu créer plusieurs compartiments dans un S3, un service de stockage d'AWS. Mais également, j'ai pu travailler sur Sagemaker, l'outil qui permet aux développeurs et aux spécialistes des données de créer, former et déployer des modèles Machine Learning rapidement. J'ai également appris à créer des plugin chrome en userscript. J'ai appris à récupérer automatiquement les images présentes sur une page web. En somme, ce stage m'a permis d'acquérir beaucoup de compétences supplémentaires, mais également des connaissances plus poussées sur les réseaux de neurones, en particulier les réseaux de neurones convolutionnels.

J'ai permis à l'entreprise d'avoir toutes les clés en main pour permettre une modération plus efficace lorsque le besoin se fera sentir. Il existe encore différentes problématiques pouvant permettre d'automatiser entièrement la modération, comme la détection d'enfants, d'alcool ou de tabac. Des problématiques que nous pourrions peut-être répondre en Janvier dans le cadre d'une thèse. Ce stage fut un vrai plus dans ma formation en tant que data scientist.

## Références

- [1] Classez et segmentez des données visuelles. <https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles>.
- [2] Convolutional neural networks for visual recognition (stanford). <http://cs231n.github.io/>.
- [3] Elo rating system. [https://en.wikipedia.org/wiki/Elo\\_rating\\_system](https://en.wikipedia.org/wiki/Elo_rating_system).
- [4] An intuitive explanation of convolutional neural networks comments feed. <https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/>.
- [5] P. M. Hossein Talebi. Nima : Neural image assessment. <https://arxiv.org/pdf/1709.05424.pdf>.
- [6] L. J. D. X. Lingyu Liang, Luojun Lin and M. Li. Scut-fbp5500 : A diverse benchmark dataset for multi-paradigm facial beauty prediction. <https://arxiv.org/pdf/1801.06345.pdf> SCUTdataset.