8/31/2023

Deep Learning

Table of Contents

[1 Introduction 1](#_Toc148266278)

[2 Literature Review 2](#_Toc148266279)

[2.1 Deep Reinforcement Learning 2](#_Toc148266280)

[2.1.1 Q-Learning 3](#_Toc148266281)

[2.1.2 Proximal Policy Optimization (PPO) 5](#_Toc148266282)

[2.1.3 Deep Q-Networks (DQN) 7](#_Toc148266283)

[3 Convolutional Neural Network (CNN) 8](#_Toc148266284)

[3.1 Residual Networks (RasNet) 9](#_Toc148266285)

[3.2 Densely Connected Networks (DenseNet) 10](#_Toc148266286)

[3.3 Visual Geometry Group Networks (VGG) 11](#_Toc148266287)

[3.4 MobileNet 12](#_Toc148266288)

[3.5 EfficientNet 13](#_Toc148266289)

[3.6 Xception 14](#_Toc148266290)

[4 Material and Methods 15](#_Toc148266291)

[4.1 Material 15](#_Toc148266292)

[4.2 Method 16](#_Toc148266293)

[4.2.1 Deep Reinforcement Learning 16](#_Toc148266294)

[4.2.2 RasNet Convolutional Neural Network (CNN) 16](#_Toc148266295)

[4.3 Flowcharts 17](#_Toc148266296)

[5 Results and Discussion 19](#_Toc148266297)

[5.1 Deep Reinforcement Learning 19](#_Toc148266298)

[5.2 RasNet CNN 21](#_Toc148266299)

[6 Conclusion 23](#_Toc148266300)

[7 References 24](#_Toc148266301)

[Figure 1: Q-Learning architecture (Awan 2022). 5](#_Toc148266710)

[Figure 2: High-level diagram of the proximal policy optimization algorithm (Anwar et al. 2022). 7](#_Toc148266711)

[Figure 3: DQN (Face 2023). 9](#_Toc148266712)

[Figure 4: Convolutional Neural Network (Shah 2022). 11](#_Toc148266713)

[Figure 5: Residual Network (Sahoo 2018). 12](#_Toc148266714)

[Figure 6: Dense neural network with 2 fully connected layers, 2 dropout layers and a decision layer with sigmoid activation (Gilitschenski et al. 2016). 13](#_Toc148266715)

[Figure 7: Visual geometry group network-16 (VGG-16) transfer learning model (Dubé and Paré 2003). 14](#_Toc148266716)

[Figure 8: Mobile Net v2 (Akay et al. 2021) 15](#_Toc148266717)

[Figure 9: Different scaling methods vs. Compound scaling (Sarkar 2021). 16](#_Toc148266718)

[Figure 10: Original Depthwise Separable Convolution (Tsang 2018). 17](#_Toc148266719)

[Figure 11: DNQ 19](#_Toc148266720)

[Figure 12: RasNet CNN 20](#_Toc148266721)

[Figure 13: DNQ confusion matrix 21](#_Toc148266722)

[Figure 14: Comparison of how many were predicted compared to the total of each class. 22](#_Toc148266723)

[Figure 15: DNQ ROC\_Curve. 22](#_Toc148266724)

[Figure 16: RasNet confusion matrix 23](#_Toc148266725)

[Figure 17: RasNet predicted vs Total. 23](#_Toc148266726)

[Figure 18: RasNet ROC\_Curve 24](#_Toc148266727)

Deep Reinforcement Learning and RasNet Convolution Neural Network for Image Classification using CIFAR-10 Dataset

# Introduction

Deep Learning and Deep Reinforcement Learning stand as a formidable subfield within the expansive domain of Machine Learning, each harbouring the potential to significantly impact a myriad of industries and applications (Awan 2022), (Goodfellow et al. 2016), (Mnih et al. 2015).

At its core, Deep Learning is a specialized branch of Machine Learning that employs artificial neural networks to tackle and decipher intricate problems. It boasts an inherent ability to learn and understand complex patterns and relationships embedded within data autonomously. Unlike traditional methods that demand manual feature engineering, Deep Learning algorithms autonomously adapt and improve through continuous interaction with data. They proficiently handle and process unstructured data, such as text and images, by automating the feature extraction process, thereby reducing the reliance on human expertise. The power of Deep Learning has been profoundly felt across various domains including, but not limited to, image recognition, natural language processing, speech recognition, and recommendation systems (Awan 2022), (Goodfellow et al. 2016), (Mnih et al. 2015).

On the flip side, Deep Reinforcement Learning (Deep RL) emerges from the confluence of Reinforcement Learning’s (RL) decision-making process and Deep Learning’s pattern recognition abilities. RL encompasses a suite of goal-driven algorithms, which are engineered to achieve complex objectives or maximize certain dimensions over a multitude of steps. Within the framework of Deep RL, deep neural networks are utilized to approximate the optimal action-value function. This function maps states to the expected total reward for each possible action, forming the basis for determining the optimal policy. Deep RL algorithms are uniquely equipped to process extensive inputs (for instance, every pixel displayed on a video game screen) and discern actions aimed at optimizing a specific objective (such as maximizing the game score) (Awan 2022), (Goodfellow et al. 2016), (Mnih et al. 2015).

The amalgamation of Deep Learning and RL within Deep RL paves the way for the development of systems capable of learning from their interactions and making informed decisions. Deep Learning bestows these systems with the capability to comprehend and interpret complex inputs, while RL empowers then to execute actions aligned with maximizing a particular objective. The versatile of Deep RL renders it a potent tool across a diverse spectrum of applications ranging from robotics and video games to natural language processing and computer vision (Awan 2022), (Goodfellow et al. 2016), (Mnih et al. 2015).

# Literature Review

## Deep Reinforcement Learning

In the expansive domain of machine learning, Deep Reinforcement Learning (Deep RL) emerges as a compelling paradigm aimed at mastering decision-making through interaction with the environment. This approach hinges on the principles of learning from trial and error, where an agent is exposed to a situation, takes actions, and receives feedback in the form of rewards or penalties. The quintessence of Deep RL lies in the agent's ability to autonomously discover the optimal strategy, or policy, that yields the maximum cumulative reward over time, even in the face of complex, high-dimensional input spaces. Deep RL amalgamates the strengths of Deep Learning, known for its prowess in unearthing intricate patterns from raw data, with Reinforcement Learning's adeptness at navigating through a labyrinth of decisions to achieve a goal. This amalgamation has birthed significant advancements in areas as diverse as robotics, autonomous vehicles, game playing, and natural language processing. Among the prominent algorithms under the Deep RL umbrella are Q-Learning, Deep Q-Networks (DQN), Proximal Policy Optimization (PPO), and Actor-Critic methods. The ensuing subsections will delve into the intricacies of Q-Learning, DQN, and Proximal Policy Optimization, elucidating their core mechanisms, applications, and the evolutionary trajectory that has positioned Deep RL as a cornerstone for developing intelligent, self-learning systems.

### Q-Learning

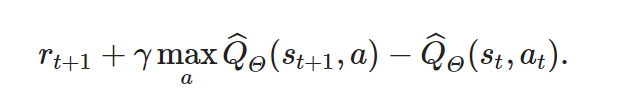
A diagram of a system

Description automatically generated

Figure : Q-Learning architecture (Awan 2022).

Q-learning, as introduced by Watkins (1989), is a fundamental model-free reinforcement learning algorithm that aims to teach agents how to choose optimal actions that yield the most cumulative reward over time in Markovian domains. Unlike model-based approaches, Q-learning does not require a model of the environment but learns from interaction with the environment. The core idea is to estimate the values of state-pairs, , which represent the total expected rewards an agent can get, starting from state , taking action , and then acting optimally onward (Watkins 1992).

At heart of Q-learning is the Q-function, defined as , which represents the expected return (cumulative discounted reward) of taking action in state and acting optimally onward (Watkins 1992). The Q-function is updated iteratively using the following rule:



Where:

* and are the current state and action,
* is the immediate reward received,
* is the discount factor which determines the agent’s consideration for future rewards,
* is the learning rate which determines to what extent newly acquired information overrides old information.

The convergence proof ensures that under certain conditions, the Q-value function will converge to the true Q-value function, , which represents the expected cumulative reward of taking action in state and acting optimally onward. The conditions for convergence include:

* All action-state pairs are visited infinitely often.
* The learning rates α satisfy certain conditions to ensure a balance between learning new information and retaining old information.

The proof leverages a constructed Markov process called the action-replay process (ARP) to show that the estimates of the Q-values converge to the true Q-values as the number of iterations goes to infinity. The ARP can be visualized as a card game. Envision each episode inscribed on a card. Collectively, these cards from an unending deck, organized with the first episode-card near the bottom, and the deck extending infinitely upward in a sequential order. The bottom card, labelled O, displays the agent’s initial values for all state-pairs . A state in the ARP, denoted , comprises a card number (or level) , along with a state from the actual process. The action set in the ARP mirrors that in the actual process (Watkins 1992).

The transition to the next state in the ARP, given the current state and action , unfolds as follows: Initially, all cards representing episodes beyond are discarded, leaving behind a finite deck. Cards are then singly removed from the top of this desk and inspected until a card is found with a starting state and action corresponding to and , say at episode . Subsequently, a biased coin is tossed, with a probability of landing heads, and of tails. A heads outcome leads to the re-enactment of the episode noted on this card; a tails outcome results in the discard of this card, and the research resumes for another matching card for and . If the search descends to the bottom card, the game transitions into a unique, absorbing state, yielding solely the reward inscribed on this card for that is . The re-enactment of the episode on the card entails the emission of the reward, , inscribed on the card followed by a transition to the next state in the ARP, where denotes the state, the actual process transitioned to in that episode. Card is then discarded. The subsequent state transition within the ARP will be orchestrated based on the remaining deck (Watkins 1992).

The significance of Q-learning lies in its simplicity, efficiency, and the ability to handle problems of sequential decision making under uncertainty without requiring a model of the environment. The convergence proof provides a solid theoretical foundation that guarantees the algorithm's ability to learn the optimal policy over time. This makes Q-learning a cornerstone in the field of reinforcement learning (Watkins 1992).

### Proximal Policy Optimization (PPO)

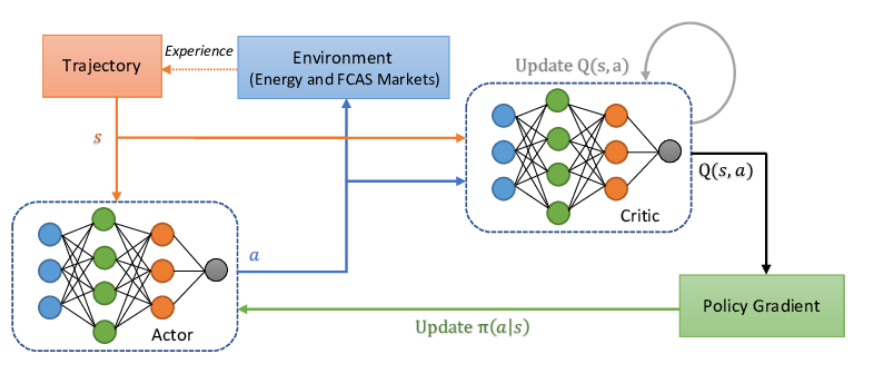


Figure : High-level diagram of the proximal policy optimization algorithm (Anwar et al. 2022).

The PPO algorithm is a class of reinforcement learning algorithms that are simpler to implement and tune and perform comparably or better than state-of-the-art approaches. Here’s high-level overview of how it works:

* **Policy definition**: In reinforcement learning, a policy refers to the strategy employed by the agent that defines a mapping from the state space to the action space. It essentially serves as a set of instructions, guiding the RL agent on which action to undertake depending on the current state of the environment (sidsen99 2022). A policy parameterized by is often denoted as where:
* is the action the agent will take,
* is the current state of the environment,
* are parameters of the policy,
* denotes the policy itself (Schulman et al. 2017).
* **Policy Evaluation**: The performance of the agent is assessed by evaluating the policy function. This is where Policy Gradient methods are utilized (sidsen99 2022).
* Policy Gradient: Represents a method wherein an agent learns the best actions to take in given states, especially when it initially lacks this knowledge. This method operates akin to a neural network architecture. Specifically, the agent computes the gradients of the output, which in this context refers to the logarithm of the probabilities associated with each action in a particular state, with respect to the environment’s parameters. These gradients are then used to adjust the policy, leading to a change in the agent’s behaviour in a way that ideally improves it performance over time (sidsen99 2022).
* **Objective Function**: Aims to update the policy at each step to minimize a cost function, while ensuring that the deviation from the previous policy remains relatively small. The objective function for PPO, denoted as , is given by:

Where:

* represents the parameters of the policy,
* denotes the empirical expectation over time,
* is the ratio of the probability of the action under the new policy to the probability of the action under the old policy,
* is an estimator of the advantage function at time ,
* is the clipped version of the ratio ,
* And is a small hyperparameter that determines the extend to which the policy can change at each update.

The objective function encourages the optimization to improve the policy, but not too drastically to prevent instability in training (sidsen99 2022).

* **Policy update**: PPO is a type of policy method that learns from the online data. A distinguishing feature of PPO is its emphasis on ensuring that the updated policy doesn’t deviate drastically from the previous policy, which aims to maintain a low variance during training. This is achieved through a specialized objective function that penalizes changes in the policy that could lead to a large deviation in behaviour.

A widely adopted implementation of PPO is through the Actor-Critic Model, which employs two Deep Neural Networks. In this setup:

* The Actor Network: It is responsible for determining the actions to be taken based on the current state of the environment.
* The Critic Network: It evaluates the actions taken by the actor estimating the expected rewards or value of the states (sidsen99 2022).

The policy update in PPO typically involves multiple steps of Stochastic Gradient Descent (SGD), often carried out on minibatches, to maximise the objective function. This process iteratively refines the policy, aiming to find an optimal policy that maximizes the expected cumulative rewards over time.

### Deep Q-Networks (DQN)

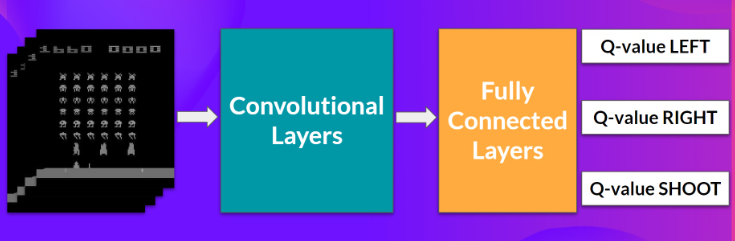
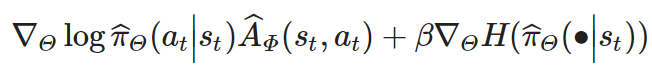


Figure : DQN (Face 2023).

DNQ utilizes a deep neural network to approximate the action-value function denoted as . Upon encountering a transition , the parameters of the neural network are adjusted to reduce the Bellman error.

To mitigate overfitting, the algorithm employs a method known as experience replay, wherein numerous transitions are captured and stored in a database. During each iteration, a selection of transitions is randomly drawn from the database to facilitate the update of the network parameters (Jonsson 2019). The asynchronous advantage Actor-Critic (A3C) algorithm maintains two estimates: one for the policy (termed as the actor, denoted by ), and another for the value function (termed as critic, denoted by ). Given a transition , it updates the policy’s parameter vector using a regularized policy gradient rule (Jonsson 2019).



This rule incorporates an estimation of the advantage function, which essentially measures the relative value of taking action in state .

A group of math symbols

Description automatically generated

Additionally, it includes a term for policy entropy to encourage exploration, with the level of regularization controlled by parameter β. The algorithm’s stability is further enhanced by utilizing n-step returns, representing the reward accumulated over consecutive transitions. Often, parameter vectors and share a certain parameter, especially in a neural network configuration where all layers except the output layers are shared between the actor and the critic. On the other hand, AlphaZero also keeps estimates for both the policy and the value function but diverges from A3C in the method of parameter updating. Instead of using the policy gradient rule, it employs a Monte-Carlo Tree Search (MCTS) to derive a target action distribution based on the empirical visitation count of each branch in the search tree, which also informs the selection of the next action . The parameters are updated through a loss function that encapsulates the difference between observed returns and estimates values, along with a regularization term controlled by β (Jonsson 2019).

# Convolutional Neural Network (CNN)

A CNN stands as a cornerstone in the realm of Deep Learning, particularly for tracking tasks within the domain of Computer Vision. This specialized neural network architecture empowers computers to interpret and analyse visual data, paving the way for numerous applications like image and video recognition, anomaly detection in medical imaging, and even autonomous driving (GeeksforGeeks 2023), (Cloud 2023), (Gurucharan 2022). Here’s a brief walkthrough of how CNN operates according to (GeeksforGeeks 2023), (Cloud 2023), (Gurucharan 2022):

* Input Layer: The journey of data through CNN begins at the Input Layer. Here, the model receives its input, with the number of neurons matching the total number of features present in the data. In the context of image processing, each pixel of the image represents a feature, and thus, the number of neurons would equal the total number of pixels.
* Convolutional Layer: As the name suggests, the convolutional layer performs convolutional operations, applying various filters to the input image to extract essential features like edges, textures, and shapes. This layer acts as the heart of the CNN, where the bulk of computational activities take place. The extracted features are then passed on to subsequent layers for further processing.
* Pooling Layer: Following the convolutional operations, the Pooling Layer steps into down sample the feature maps, essentially reducing their dimensions. This down sampling helps in cutting down the computational load and aids in making the detection of features invariant to scale and orientation changes.
* Fully Connected Layer: After several rounds of convolutional and pooling, the data reaches the Fully Connected layer. In this segment, every neuron connects to every neuron in the previous layer, interpreting the computed features to make the final predictions. The data is then funnelled through activation functions like sigmoid or SoftMax, transforming the outputs into probability scores for each class.
* Backpropagation: The last crucial phase in training of a CNN is Backpropagation. Post the forward pass, the model computes the error by contrasting the predictions against the true labels using a loss function. Backpropagation then kicks in, traversing back through the network, adjusting the internal parameters (weights and biases) to minimize the error, and hence, improving the model’s performance.

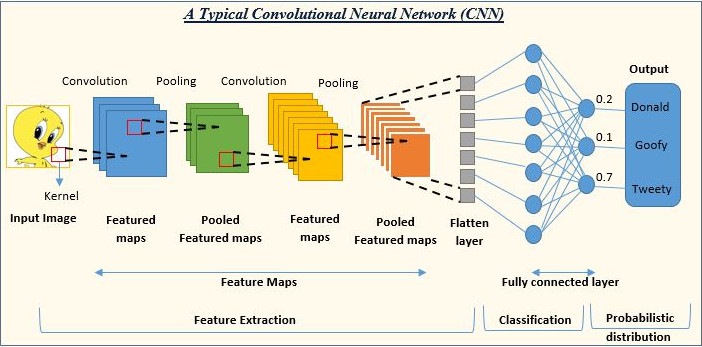


Figure : Convolutional Neural Network (Shah 2022).

Through the meticulous interplay of these layers and processes, CNNs learn to discern patterns in visual data, opening a vista of possibilities in the artificial intelligence applications (GeeksforGeeks 2023), (Cloud 2023), (Gurucharan 2022). The following subsections will dive deep into the variants of CNN.

## Residual Networks (RasNet)

RasNets are a class of deep learning models that are introduced the innovative gradient problem prevalent in very deep networks. RasNets have profoundly impacted the deep learning domain, especially in visual data analysis tracks. Crafted by Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun, RasNets clinched victory in the ImageNet 2015 competition. Here’s a simplified explanation of how RasNets operate:

* Input Layer: This layer receives our data. In image-related tasks, the neuron count in this layer equals the total pixel count in the image.
* Convolutional Layer: This layer employs filters on the input image to extract vital features, and where the bulk of computation takes place.
* Residual Block: This pivotal component of a RasNet comprises several layers (such as convolutional layers, batch normalization layers, and ReLU layers), alongside a skip connection that facilitates the addition of the block’s input to its output.
* Identity Shortcut Connection: This connection executes an identity mapping, enabling the input to bypass one or more layers to be directly added to the output. This “residual” aspect of the RasNet helps counteract the vanishing gradient issue.
* Activation function: Post each convolution operation and the identity shortcut connection, an activation function like ReLU is applied.
* Pooling Layer: Utilized to diminish the spatial dimensions of the data, this layer in turn reduces the parameter count and computation load in the network.
* Fully Connected Layer: Typically positioned at the network’s end, this layer delivers the final classification result.
* Softmax or Sigmoid Function: In the final step, a softmax or sigmoid function is utilized to convert the raw class scores from the fully connected layer into probabilities.

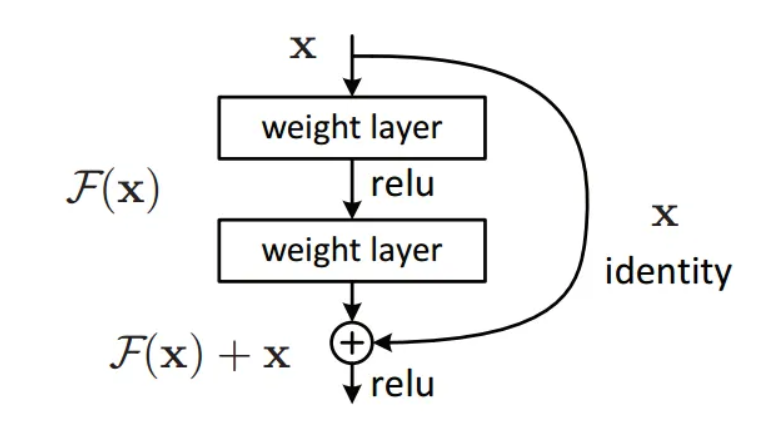


Figure : Residual Network (Sahoo 2018).

## Densely Connected Networks (DenseNet)

DenseNets represent a variant of Convolutional Neural Networks (CNNs) recognized for their depth, accuracy, and training efficiency. Distinctively, DenseNets exhibit a unique architecture where every layer is interconnected in a feed-forward manner, contrasting with traditional CNNs where layers are sequentially linked. Specifically, instead of having just L connections for L layers as in typical CNNs, DenseNets boast L(L+1)/2 direct links. This means that the output from a given layer is used as input for all succeeding layers. This dense connectivity offers numerous benefits. It mitigates the vanishing-gradient challenge, bolsters feature transmission, promotes the reuse of features, and remarkably decreases the required parameters. When benchmarked against notable object recognition tasks like CIFAR-10, CIFAR-100, SVHN, and ImageNet, DenseNets not only outperformed many state-of-the-art models but also demanded fewer computational resources to achieve top-tier results.

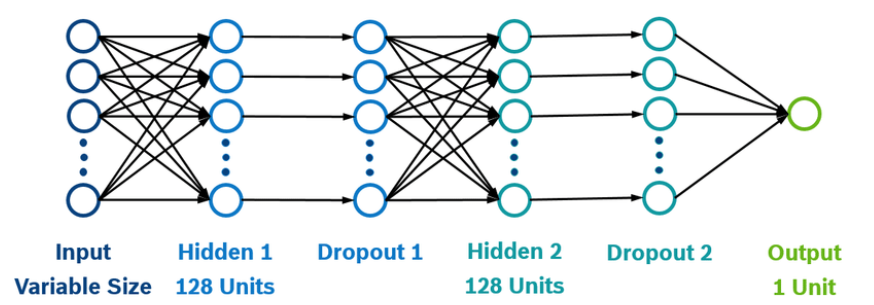


Figure : Dense neural network with 2 fully connected layers, 2 dropout layers and a decision layer with sigmoid activation (Gilitschenski et al. 2016).

## Visual Geometry Group Networks (VGG)

The VGG Networks or VGGNet, represent a Convolutional Neural Network (CNN) design pioneered by the University of Oxford's Visual Geometry Group. The hallmark of VGGNet's design is its straightforward architecture, predominantly featuring 3x3 convolutional layers stacked consecutively. This design choice is grounded in the belief that several 3x3 convolutions can mimic the effect of larger receptive fields, like 5x5 or 7x7, but with reduced parameters and heightened non-linearities. Notably, VGGNet has two primary versions: VGG16, comprising 16 layers (13 convolutional and 3 fully connected), and VGG19 with 19 layers (16 convolutional and 3 fully connected). Both have demonstrated prowess in object recognition tasks. In the 2014 ImageNet Large Scale Visual Recognition Competition (ILSVRC), VGGNet clinched the position of first runner-up in the classification category. Despite evolving models in the deep learning landscape, VGGNet's enduring appeal lies in its capacity for robust feature extraction from images.

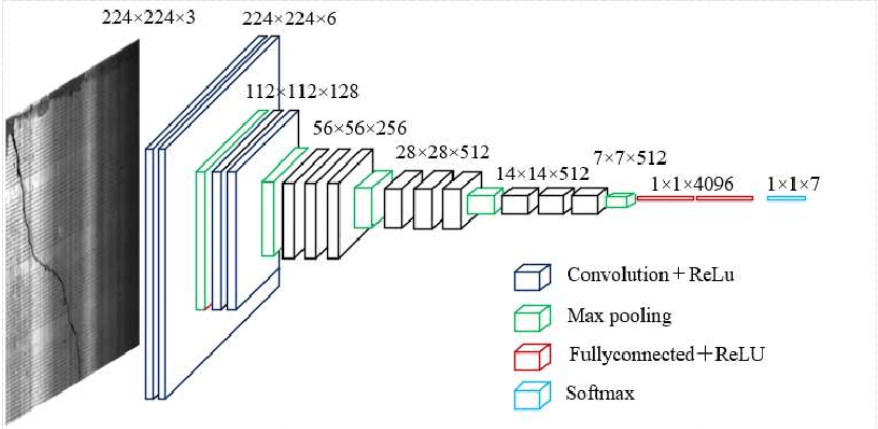


Figure : Visual geometry group network-16 (VGG-16) transfer learning model (Dubé and Paré 2003).

## MobileNet

MobileNet, a CNN model developed and open-sourced by Google, aims to bring efficiency and computational simplicity to mobile vision applications. Designed to be lightweight while maintaining robust performance, MobileNet has found extensive application in various domains, such as object detection, fine-grain classification, and face attributes and localization. At the core of the MobileNet architecture is depth-wise separable convolution, a mechanism that involves two distinctive layers: depth-wise and point-wise convolution. The former applies a single filter to each input channel, contrasting standard convolutions that apply filters across all input channels, while latter employs a 1 x 1 convolution to compute a linear combination of the output from the depth wise convolution. MobileNetV2, a subsequent iteration in the MobileNet series, continues to leverage CNNs to identify image features, discerning shapes and matches among objects. Collectively, the MobileNet models are crafted to optimize performance, particularly on mobile and edge devices, ensuring applicability and utility in real-world, on-the-go technology.

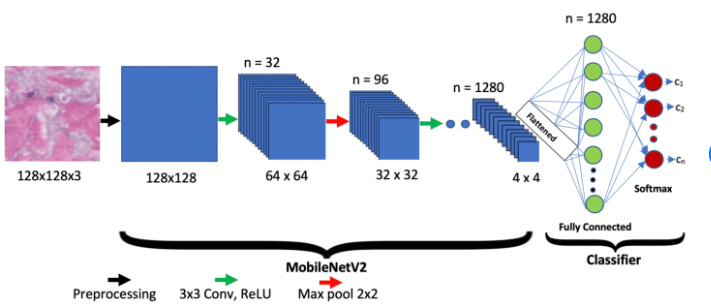


Figure : Mobile Net v2 (Akay et al. 2021)

## EfficientNet

EfficientNet, a robust CNN architecture has been engineered to deliver a compelling blend of precision and efficiency. This framework utilizes a mechanism known as compound scaling, which methodically enhances models, ensuring both simplicity and efficacy. The creators of EfficientNet advocated for the model’s expansion to enhance its accuracy and efficiency, introducing a method that cohesively and uniformly scales each dimension through a fixed set of scaling coefficients, as opposed to arbitrary scaling of width, depth, or resolution. Through this scaling approach, complemented by AutoML, they devised seven distinct models, each surpassing the benchmark accuracy and efficiency of prevalent CNNs. The fundamental premise of the compound scaling method lies in maintaining a consistent ratio while scaling dimensions such as width, depth, and resolution. This methodology posits that larger input images necessitates networks with additional layers to expand the receptive field and additional channels to discern more subtle patterns on the larger image. EfficientNet has demonstrated enhanced model efficiency and precision compared to prior CNN models like MobileNet and RasNet, improving ImageNet accuracy by approximately 1.4% and 0.7% respectively, relative to other arbitrary scaling methods. Furthermore, a subset of EfficientNet models, dubbed EfficientNet-eLite, designed explicitly for edge devices like low-power IoT devices and wearables such as smart glasses and watches, has been developed, with the smallest variant surpassing the best and smallest existing MnasNet by being 1.46x more lightweight and achieving 0.56% higher accuracy.

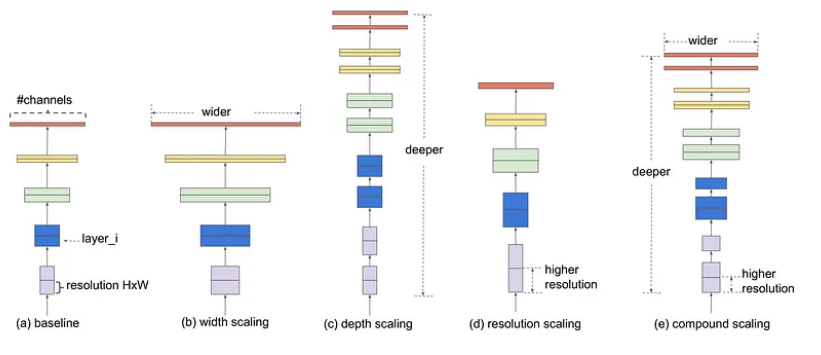


Figure : Different scaling methods vs. Compound scaling (Sarkar 2021).

## Xception

Derived from “Extreme Inception,” represents a profound architecture in the realm of deep CNNs, specifically emphasizing the utilization of Depthwise Separable Convolutions. The architecture was pioneered by Francois Chollet, a prominent figure at Google, Inc., and the architect behind Keras. Whereas the Xception, a deep neural network (DNN), employs a design laden with repeating units known as inception modules, each layer meticulously extracts features, which are then methodically stacked. Despite the strategic layering, deep networks often grapple with overfitting and compounded with sequential convolutional operations, present a computationally taxing training process. The inception module uniquely executes multiple transformations on identical inputs, subsequently combining the outputs, thereby granting the model the autonomy to discern and quantify feature utilization. However, computational efficiency remains a challenge due to the convolutions occurring both spatially and across depth. Xception amplifies the inception principles, reversing the process utilized in Inception by employing various types of filters from each depth space on the original input compressed using 1x1 convolutions. Notably, a pre-trained version of the Xception network, proficient in classifying images into 1000 project categories, has been trained on over a million images from the ImageNet database, offering robust applicability in object detection and categorization.

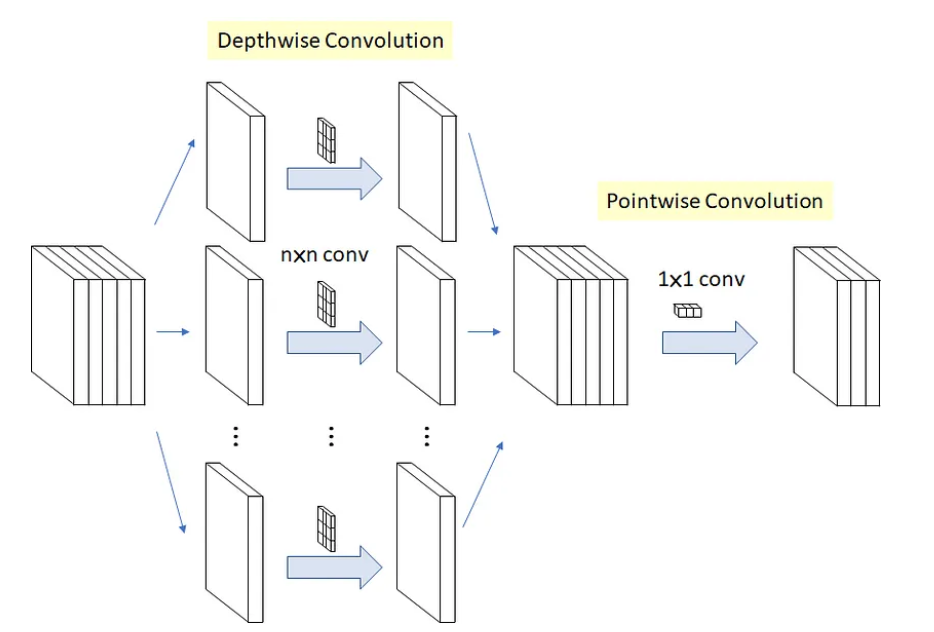


Figure : Original Depthwise Separable Convolution (Tsang 2018).

# Material and Methods

This project hinges on devising and implementing two separate models: a Deep Reinforcement Learning (RL) specifically Deep Q-Network and a Convolutional Neural Network (CNN), specifically RasNet for classification of images from CIFAR-10 dataset, exploring both the realms of Deep RL and conventional image classification independently.

## Material

The CIFAR-10 dataset serves as the foundational material for this project, comprising 60 000 colour images, each of 32x32 pixels, neatly distributed across 10 distinct classes, ensuring a balanced representation with 6 000 images per class. The pixel values, ranging from 0 to 255 across three colour channels (Red, Green, and Blue), act as the independent variables, while the target variable in the image class, with possible labels including airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

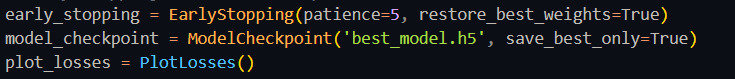
## Method

### Deep Reinforcement Learning

The project employs a Deep Q-Network (DNQ) to traverse the CIFAR-10 dataset, with the network interacting with its environment (the images) and determining optimal classifications through a reward or punishment mechanism inherent to Reinforcement Learning. The network parameters, such as learning rate, gamma (discount factor), and epsilon (exploration-exploitation trade-off), were meticulously varied to observe influences on convergence and learning stability. The dataset was partitioned adhering to an 80/20 train/test split, ensuring ample data for training while reserving a robust sunset for validation. The DNQ ingests an image, determining a classification (action) based on its present understanding, receiving a reward or penalty and subsequently refining its future actions through updating Q-values, iterating this process until convergence or for a preset number of episodes.

### RasNet Convolutional Neural Network (CNN)

Concurrently, the project explores the application of a RasNet CNN, leveraging supervised learning to navigate the classification of images from the CIFAR-10 dataset. The model learns by minimizing the discrepancy between its predicted class probabilities and actual classes, utilizing parameters like the number of layers, kernel size, activation function, loss function, and optimizer, adjusted to explore their impact on model performance. The data was similarly split into an 80/20 ratio for training and testing, respectively. The RasNet model processes an image through convolutional activation, and pooling layers, converting it into feature maps, which then traverse additional layers to predict class probabilities. The error (loss) between predictions and actuals is calculated, propagated back through the network via backpropagation to adjust weights, and this process is iterated, either for a predefined number of epochs or until the early stopping conditions are met.



## Flowcharts

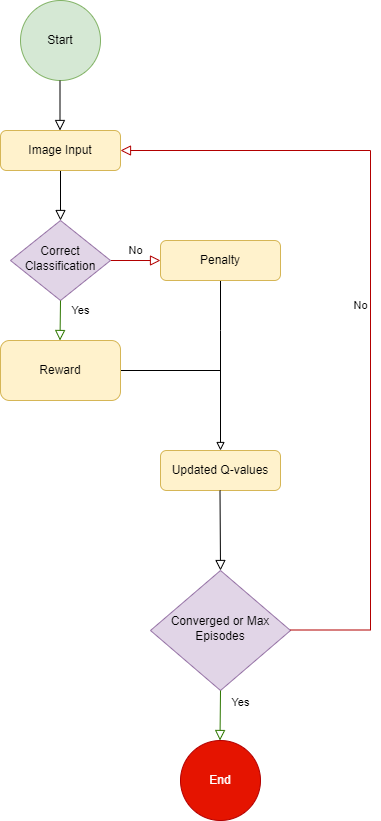


Figure : DNQ

A diagram of a company

Description automatically generated with medium confidence

Figure : RasNet CNN

# Results and Discussion

## Deep Reinforcement Learning

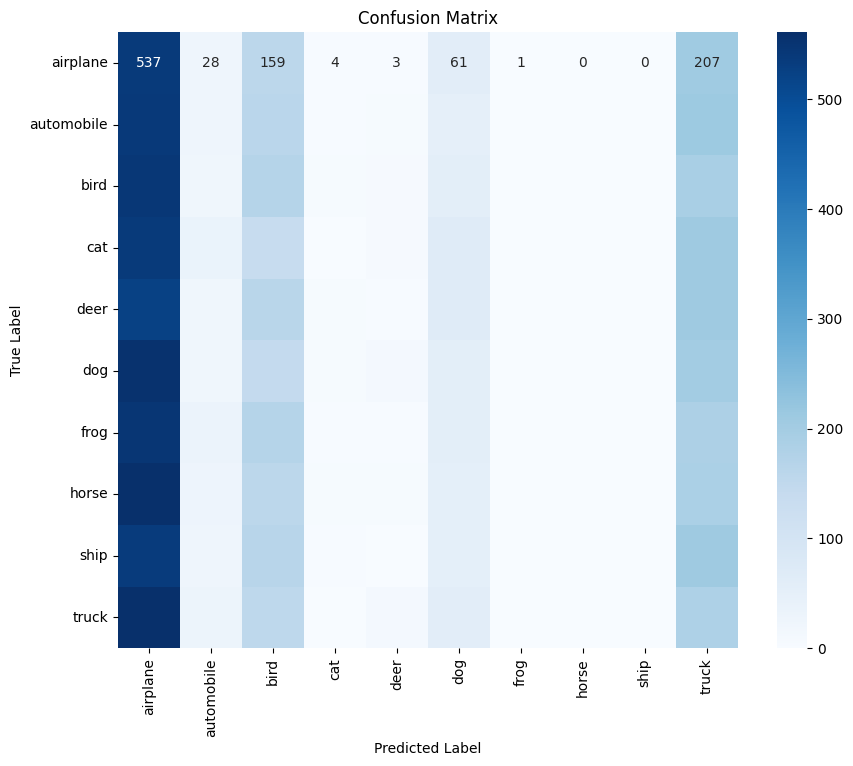


Figure : DNQ confusion matrix

Figure : Comparison of how many were predicted compared to the total of each class.

Figure 13 and 14 illustrates the experimental results obtained from the Deep Q-Network (DQN) model applied to the CIFAR-10 image classification task. A noticeable struggle to achieve higher accuracy percentages is depicted, reflecting the model's challenge in effectively navigating through the action spaces and managing the complexity inherent in image data. This visual representation underscores the intricacies and potential pitfalls of applying reinforcement learning models, like DQN, to high-dimensional, complex tasks outside of their traditionally successful domains, such as game playing. The airplane class was the most predicted class compared to the other classes with the value of 537.

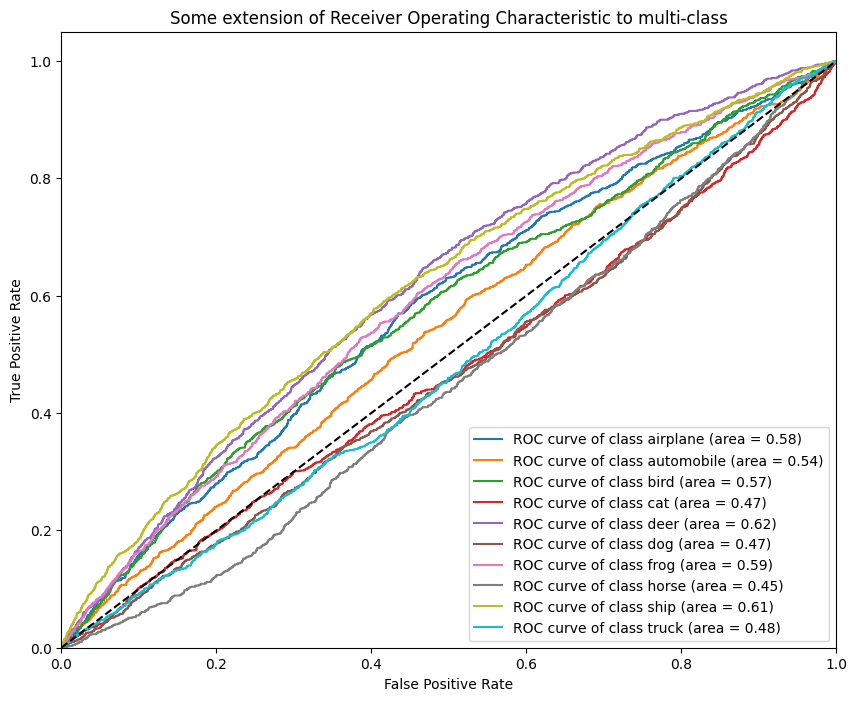


Figure : DNQ ROC\_Curve.

The Receiver Operating Characteristic (ROC) curve represent the performance of the DQN model on the CIFAR-10 image classification task. The ROC curve, which illustrates the trade-off between the true positive rate and false positive rate across different threshold settings, exhibits considerable areas where the model struggles to discriminate between the classes effectively. The model’s 9% accuracy is highlighted by the curve's deviation from the optimal top-left corner, indicating challenges in achieving both high sensitivity and specificity in class predictions. This visual insight into the model’s predictive capabilities underscores the complexities and hurdles faced by DQN in managing high-dimensional image classification.

## RasNet CNN

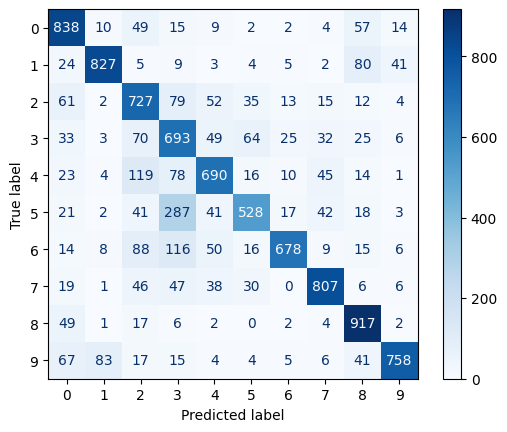


Figure : RasNet confusion matrix.

Figure : RasNet predicted vs Total.

Displayed above are the outcomes from implementing the ResNet model, revealing a stark contrast against the DQN model in effectively classifying images from the CIFAR-10 dataset. The pronounced success, evidenced by a high accuracy of 91%, underscores the model's aptitude in managing the intricacies of image data, extracting pertinent features, and successfully navigating the classification task. The depth of the model, alongside its innovative use of skip connections, facilitates an adept learning from the image data, capturing hierarchical features and contributing to its formidable performance.

A graph of a function

Description automatically generated with medium confidence

Figure : RasNet ROC\_Curve.

The ResNet model's performance and ROC curve, as visualized above, showcase a robust capability in managing the CIFAR-10 classification task. With a remarkable 91% accuracy, the model not only effectively classifies images but also maintains a strong balance between sensitivity and specificity, as depicted by the ROC curve. The curve, representing the true positive rate against the false positive rate, leans towards the top-left corner of the plot, indicating an adept ability to discriminate between classes while minimizing misclassification. The ResNet model, through its architectural depth and skip connections, seemingly navigates the nuances and challenges of image data effectively, resulting in a powerful, accurate predictive model.

In this comparative analysis between the Deep Q-Network (DQN) and ResNet models, the latter emerged distinctly superior, boasting a compelling 91% accuracy against the 9% achieved by the DQN. The disparities in performance can be attributed to several factors.

Concerning the DQN model, the modest 9% accuracy illuminates potential difficulties in grappling with the image classification task, hinting at probable hurdles in adeptly navigating through the action space and extracting meaningful learning from the environment. The sophistication of image data, characterized by its high-dimensionality and intricacy, might have presented formidable challenges in formulating state representations, and making decisive actions within the DQN framework.

Conversely, the ResNet model, with its impressive 91% accuracy, amplifies the efficacy of convolutional layers in proficiently handling image data by systematically and effectively extricating relevant features. The architecture of ResNet, renowned for its depth and innovative use of skip connections, likely facilitated an adept learning from image data. This was achieved by capturing hierarchical features and enabling the flow of gradients through the network, contributing significantly to its eminent performance.

In sum, the contrast in performances underscores the criticality of selecting model architectures that are intrinsically aligned with the nature of the data and the task, ensuring optimal extraction of patterns and, consequently, robust predictive performances.

# Conclusion

In conclusion, this project undertook a meticulous exploration into the realms of Deep Learning, particularly diving into the capacities of Deep Q-Networks (DQN) and ResNet Convolutional Neural Network (CNN) in the domain of image classification, employing the CIFAR-10 dataset. The evident disparity in performance between the two models, with ResNet distinctly outperforming DQN, has unveiled critical insights into model selection and its impact on predictive accuracy in image classification tasks.

ResNet, with its profound architectural depth and intelligent utilization of skip connections, demonstrated a remarkable capability in managing high-dimensional image data, securing an impressive 91% accuracy. On the other hand, DQN, although a potent model in certain reinforcement learning contexts, may have faced inherent challenges in dealing with the complexity and nuances of image data, resulting in a modest 9% accuracy.

This divergence in model efficacy underscores the importance of aligning model architecture with the specificities and demands of the data and task at hand. Consequently, while DQN has demonstrated proficiency in various domains, its application to image classification, especially when juxtaposed against a potent CNN variant like ResNet, may not be the most optimal choice.

The findings from this exploration not only shed light on the imperative of strategic model selection but also fortify our understanding of the robust capabilities and limitations that different neural network architectures present in navigating through diverse problem spaces within the expansive landscape of machine learning. This endeavour, thus, not only contributes to our academic and practical knowledge but also paves the way for future research, where the exploration of various model architectures across different tasks and data types can be further delved into, facilitating the continuous expansion and refinement of the field of Deep Learning.

# References

Akay, M., Du, Y., Sershen, C. L. and Wu, M. (2021) 'Deep Learning Classification of Systemic Sclerosis Skin Using the MobileNetV2 Model', *Research Gate,* 99(1).

Anwar, M., Wang, C., Nijs, F. d. and Wang, H. (2022) 'Proximal Policy Optimization Based Reinforcement Learning for Joint Bidding in Energy and Frequency Regulation Markets', 2023(02 October),

Awan, A. A. (2022) 'An Introduction to Q-Learning: A Tutorial for beginners', [online], available: <https://www.datacamp.com/tutorial/introduction-q-learning-beginner-tutorial> [Accessed 28 September 2023].

Cloud, I. (2023) 'What are convolutional neural networks?', [online], available: <https://www.ibm.com/topics/convolutional-neural-networks> [Accessed 02 October 2023].

Dubé, L. and Paré, G. (2003) 'Rigor in information systems positivist case research', *current practices, tre. MIS Quarterly,* 4(27), 597-635.

Face, H. (2023) 'The Deep Q-Network (DNQ)', [online], available: <https://huggingface.co/learn/deep-rl-course/unit3/deep-q-network> [Accessed 15 October 2023].

GeeksforGeeks (2023) 'Introduction to Convolution Neural Network', *GeeksforGeeks,* 2023(03 October),

Gilitschenski, I., Siegwart, R., Nieto, J. I., Behrendt, K., Mielenz, H. and Volz, B. (2016) 'A data-driven approach for pedestrian intention estimation', in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*,

Goodfellow, I., Bengio, Y. and Courville, A. (2016) *Deep Learning,* USA:Massachesetts Institute of Technology.

Gurucharan, M. (2022) 'Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network', *upGrad,* 2023(02 October),

Jonsson, A. (2019) 'Deep Reinforcement Learning in Medicine', *National Library of Medicine,* 5(1), 18-22.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Reidmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S. and Hassabis, D. (2015) 'Human-level control through deep reinforcement learning', *nature,* (519), 529-533.

Sahoo, S. (2018) 'Residual blocks — Building blocks of ResNet', *Towards Data Science,*

Sarkar, A. (2021) 'Understanding EfficientNet — The most powerful CNN architecture', *Medium*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O. (2017) 'Proximal Policy Optimization Algorithms', *Cornell University*.

Shah, S. (2022) 'Convolutional Neural Network: An Overview', *Analytics Vidhya,* 2023(02 October),

sidsen99 (2022) 'A Brief Introduction to Proximal Policy Optimization', [online], available: <https://www.geeksforgeeks.org/a-brief-introduction-to-proximal-policy-optimization/> [Accessed 02 October 2023].

Tsang, S.-H. (2018) 'Review: Xception — With Depthwise Separable Convolution, Better Than Inception-v3 (Image Classification)', *Medium*.

Watkins, C. J. C. H. (1992) 'Q-Learning', *Kluwer Academic Publishers,* (8), 279-292.