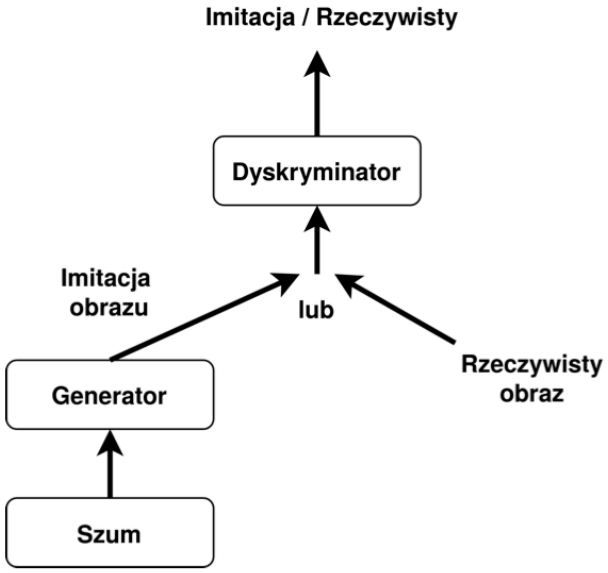
# *Lab 8 - GAN*

1. **Wstęp**

Generative Adversarial Networks (GAN) to model stosowany do generowania danych. Znajduje zastosowanie w tworzeniu obrazów, transformacji stylów czy generowaniu treści. GAN opiera się na rywalizacji dwóch sieci neuronowych, co umożliwia odwzorowywanie rozkładów danych i tworzenie nowych przykładów.

2. **Architektura**

GAN składa się z dwóch sieci neuronowych: Generatora i Dyskryminatora, które są trenowane jednocześnie w ramach treningu adwersarialnego (ang. *adversarial training*).




- **Generator**: Ta sieć przyjmuje losowy szum jako dane wejściowe i generuje dane (np. obrazy). Jej celem jest tworzenie danych jak najbardziej zbliżonych do danych rzeczywistych.
- **Dyskryminator**: Ta sieć przyjmuje dane rzeczywiste oraz dane wygenerowane przez Generator jako dane wejściowe i stara się odróżnić jedno od drugiego. Wypisuje prawdopodobieństwo, że dane są prawdziwe, realizując zadanie klasyfikacji – wskazania, czy obraz jest imitacją, czy rzeczywisty.

Podczas treningu Generator stara się generować dane, których Dyskryminator nie będzie w stanie odróżnić od danych rzeczywistych, podczas gdy Dyskryminator stara się coraz lepiej rozróżniać dane prawdziwe od fałszywych. Obie sieci rywalizują ze sobą w grze: Generator dąży do tworzenia przekonujących fałszywych danych, a Dyskryminator stara się rozróżnić te prawdziwe od fałszywych. Ten proces prowadzi do stopniowego tworzenia przez Generator coraz lepszych danych.

## 3. Trenowanie GAN

Podczas treningu, GAN przyjmuje dwa rodzaje danych wejściowych: losowy szum oraz dane wejściowe, które nie są etykietowane. Korzystając z tych dwóch danych, generuje dane, które przypominają dane wejściowe. Ponieważ wszystkie dane wejściowe do GAN nie mają etykiet, GAN jest rodzajem **nienadzorowanego** uczenia.

**Generator** przyjmuje wektor losowego szumu, zwykle próbkowany z rozkładu normalnego lub jednostajnego. Wektor ten służy jako punkt wyjścia do generowania danych. Generator aktualizuje swoje wagi na podstawie informacji zwrotnej od dyskryminatora. Jeśli dyskryminator poprawnie rozpozna wygenerowane dane jako fałszywe, generator "dostosowuje swoje wagi, aby w kolejnej iteracji wygenerować bardziej “przekonujące” dane i oszukać dyskryminator.

Problemy z trenowaniem generatora:

- **Zawężenie Modelu** (ang. *model collapse*) - Jest to sytuacja, w której generator generuje ograniczoną różnorodność wyników, mimo że jest trenowany na zróżnicowanych danych. Jest to jak sytuacja, w której generator odkrywa jeden konkretny „typ” fałszywych danych, z którym dyskryminator ma trudności, a następnie generuje tylko ten typ danych. Można to złagodzić poprzez wprowadzenie technik regularizacji, funkcji straty promujących różnorodność lub stosowanie różnych metod treningowych.
- **Niestałość Treningu -** GANy są ogólnie znane z tego, że są trudne do trenowania, a trening generatora stanowi szczególne wyzwanie. Wynika to z faktu, że krajobraz optymalizacji obejmuje dwa modele, które są trenowane jednocześnie w dynamicznym środowisku.

**Dyskryminator** - zwraca wartość skalarnego pomiędzy 0 a 1, reprezentującą prawdopodobieństwo, że dane wejściowe są prawdziwe (zazwyczaj funkcją aktywacyjną jest sigmoid). Wartość bliska 1 sugeruje, że próbka jest prawdopodobnie prawdziwa, podczas gdy wartość bliska 0 sugeruje, że jest prawdopodobnie fałszywa.
Dla danych obrazowych Dyskryminatory często wykorzystują sieci CNN. Dyskryminator GAN jest zazwyczaj prostszy niż konwencjonalne sieci CNN, ponieważ jego złożoność jest związana z połączeniem z Generatorem.

Problemy z trenowaniem dyskryminatora:

- **Zbyt silny Dyskryminator -** jeśli dyskryminator staje się zbyt silny zbyt szybko, może zawsze udzielać bardzo pewnych odpowiedzi (bliskich 0 lub 1) dla dowolnego wejścia, co utrudnia Generatorowi naukę.
- **Słaby Dyskryminator -** jeśli dyskryminator jest zbyt słaby, Generator może nie otrzymywać znaczących informacji zwrotnych, które pozwoliłyby mu się poprawić.

## 4. Funkcje straty

Zarówno generator, jak i dyskryminator zazwyczaj są oparte na binarnej entropii krzyżowej jako funkcji straty:

$$L(y, p) = -(y \log(p) + (1 - y) \log(1 - p))$$

Gdzie $y$ to rzeczywista wartość etykiety (0 lub 1), a $p$ to przewidywane prawdopodobieństwo przynależności do klasy 1.

Funkcja straty dyskryminatora składa się z dwóch komponentów: pierwszy odpowiada za optymalizację klasyfikacji rzeczywistych próbek, a drugi penalizuje błędną klasyfikację próbek generowanych przez generator.

$$L_D = -\frac{1}{2}\mathbb{E}_{x \sim p_{\text{data}}(x)} \left[\log(D(x))\right] - \frac{1}{2}\mathbb{E}_{z \sim p_z(z)} \left[\log(1 - D(G(z)))\right]$$

Gdzie:

$$\mathbb{E}_{x \sim p_{\text{data}}(x)}[f(x)] \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

$$\mathbb{E}_{z \sim p_z(z)}[f(z)] \approx \frac{1}{M}\sum_{i=1}^{M} f(z_i)$$

$x_i$ to próbki z rzeczywistego zbioru danych, $N$ to liczba próbek z rzeczywistego zbioru danych, $z_i$ to próbki z rozkładu szumu, a $M$ to liczba próbek z rozkładu szumu.

Funkcja straty generatora w GAN minimalizuje prawdopodobieństwo, że dyskryminator poprawnie odróżni fałszywe próbki wygenerowane przez generator od rzeczywistych próbek:

$$L_G = -\frac{1}{2}\mathbb{E}_{z \sim p_z(z)} \left[\log(D(G(z)))\right]$$

Łączna funkcja straty GAN, często nazywana funkcją straty minimax, jest połączeniem strat dyskryminatora i generatora, jest używana, aby wspólnie trenować generator i dyskryminator:

$$L_{\text{GAN}} = \min_G \max_D L_D + L_G$$

Do funkcji straty generatora często dodaje się penalizację gradientu (Gradient Penalty, GP) w celu stabilizacji gradientów podczas uczenia (zanikający/eksplodujący gradient). Polega to na dodaniu do funkcji straty dodatkowego składnika, który wymusza, aby normy gradientów dyskryminatora względem danych wejściowych były bliskie określonej wartości.

Więcej na temat funkcji straty w GAN można znaleźć tutaj.

## 5. CGAN oraz DAGAN

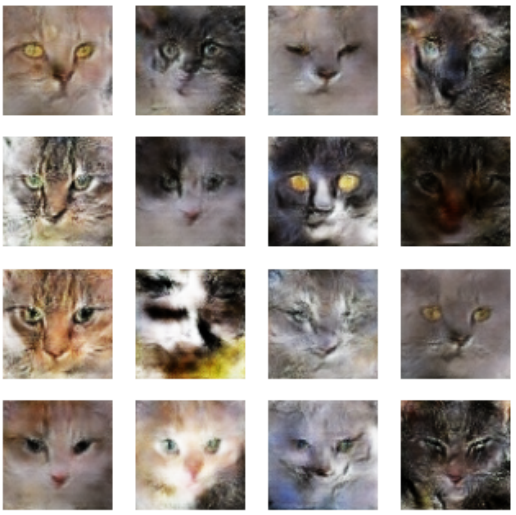
**DAGAN** (ang. *Dynamically Adaptive GAN*) to rozszerzenie GAN, które dynamicznie dostosowuje architekturę generatora i dyskryminatora w trakcie treningu, aby lepiej odwzorować rozkład danych. Więcej na temat DGAN.

**CGAN** (*Conditional Gan*) to wariant GAN, który generuje próbki na podstawie dodatkowych warunków, takich jak etykiety klas lub inne atrybuty. Więcej na temat CGAN.

## 6. Zadanie

Na podstawie danych zawierających zdjęcia twarzy kotów (znajdują się na Teams) opracuj sieć GAN do generowania nowych obrazów kotów.

*Przykłady wygenerowanych obrazów:*



*Protips:*

- *Jako początkowy model można wykorzystać ten model po odpowiednim dostosowaniu jego rozmiarów, normalizacji itd., a następnie rozwijać go według potrzeb.*