GAN - sztuczna inteligencja generująca sztukę

Temat projektu

GAN to skrót od Generative Adversarial Networks, co w wolnym tłumaczeniu oznacza generatywne sieci współzawodniczące. Jest to sposób tworzenia sztucznej inteligencji, który sprawia, że komputery mogą tworzyć różnego rodzaju obrazy ludzi, przedmiotów, czy scen, które naprawdę nigdy nie istniały, lub modyfikować obrazy istniejących już obiektów poprzez dodanie im niewystępujących wcześniej cech. Celem projektu jest wykorzystanie GAN do generacji obrazów, które swoim stylem będą odpowiadały twórczości wybranych artystów (Monet, Van Gogh, Cezanne). Inspiracją jest treść zadania z platformy Kaggle (https://www.kaggle.com/c/gan-getting-started). Autorzy zadania zasugerowali dwa różne podejścia do postawionego problemu. Pierwsze z nich polega na wykorzystaniu DCGAN i generacji obrazów od podstaw. Drugie rozwiązanie opiera się na zastosowaniu CycleGAN. W tym wariancie zadania istniejące już zdjęcia przekształcane są w taki sposób, by jak najwierniej oddać styl malarstwa charakterystyczny dla danego artysty, uniemożliwiając tym samym rozpoznanie, czy wyjściowy obraz to oryginalne dzieło, czy może wynik nałożonej transformacji. Podczas realizacji projektu zdecydowano się na implementację drugiego wariantu opisanego wyżej zadania.

Zbiory danych

Platforma Kaggle zapewnia użytkownikom dostęp do zestawów danych, których można użyć podczas pracy nad problemem. Dostępne zestawy z twórczością docelowych artystów to:

1. 300 obrazów Monet'a (zestaw z Kaggle)
2. 1074 obrazów Monet'a (zestaw wykorzystany w referencyjnej implementacji CycleGAN)
3. 401 obrazów Van Gogh'a
4. 584 obrazów Cezanne'a

Obrazy są dostępne w formacie jpeg lub tfrec. Piąty zestaw stanowi 7028 zdjęć i posłużą one do zweryfikowania efektów osiągniętych podczas procesu uczenia maszynowego. Celem zadania jest ich modyfikacja w taki sposób, by wiernie swoim wyglądem naśladowały styl malarstwa danego artysty.

Omówienie problemu

Na generatywną sieć współzawodniczącą składają się dwie niezależne sieci neuronowe - dyskryminator i generator. Pierwsza z nich uczona jest rozpoznawania obrazów i trenowana pod kątem dostrzegania kolorów, cech charakterystycznych oraz indywidualnego sposobu prowadzenia pędzla, które definiują unikatowy styl danego malarza. Drugą sieć wykorzystuje się natomiast w roli generatora dzieł sztuki, który wzorując się na właściwościach oryginałów, stara się stworzyć tysiące obrazów we wspomnianym wcześniej, unikatowym dla danego artysty stylu.

Model generujący

Model generujący jest zaprojektowany tak, aby mapować losowy wektor na próbkę z przestrzeni danych, na przykład na obraz. Wektor wejściowy jest losowany z pewnego rozkładu normalnego. Celem generatora jest estymacja rozkładu danych treningowych (czyli np. obrazów), tak aby z tego rozkładu można było generować próbki, które przypominają te dane treningowe.

Model klasyfikujący

Model klasyfikujący, inaczej zwany dyskryminatorem, jako wejście przyjmuje próbkę z przestrzeni danych i udziela binarnej odpowiedzi: prawdziwy lub fałszywy. Próbki z przestrzeni danych, które dostaje na wejście dyskryminator, pochodzą zarówno ze zbioru treningowego, jak i zbioru wygenerowanego przez generator.

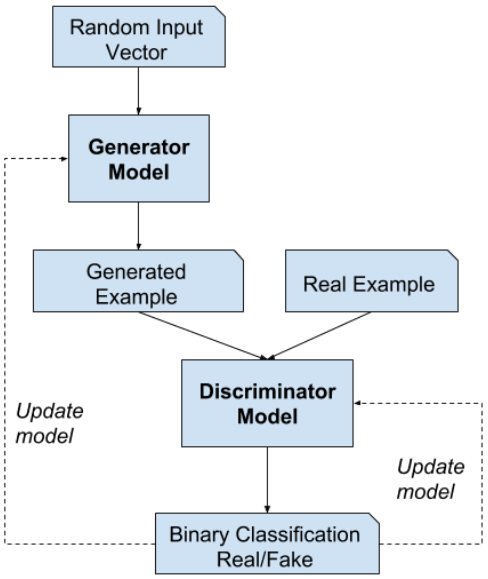
Połączenie modelu generującego oraz klasyfikującego

Obydwa modele, generator i dyskryminator, są trenowane jednocześnie. Generator produkuje kilka próbek z przestrzeni danych i one wraz z prawdziwymi próbkami ze zbioru treningowego są dostarczane jako wejście dla dyskryminatora, które klasyfikuje je jako prawdziwe (pochodzące ze zbioru treningowego), lub fałszywe (wygenerowane przez generator).

Dyskryminator jest ulepszany, aby lepiej spełniać zadania klasyfikacji w następnych krokach. Generator również jest ulepszany na podstawie tego jak dobrze (lub nie dobrze) wygenerowane próbki oszukały dyskryminator.

W ten sposób dwa modele współzawodniczą ze sobą grają w grę o sumie zerowej. W tym przypadku określenie “suma zerowa” oznacza, że jeśli dyskryminator pomyślnie identyfikuje prawdziwe lub fałszywe próbki, to nie jest aktualizowany, a parametry modelu generującego są ulepszane. Tak samo w drugą stronę - jeśli generatorowi uda się oszukać dyskryminator, to pierwszy z nich nie jest aktualizowany, a drugi ulepsza swoje parametry.

W nieskończoności, generator produkuje za każdym razem idealne próbki ze zbioru danych, a dyskryminator za każdym razem nie jest w stanie określić różnicy i zwraca wynik 50% że prawdziwy, 50%, że fałszywy.



Podsumowując, w trakcie procesu uczenia obydwa modele podnoszą swoje umiejętności - jeden system AI stara się skopiować twórczość malarza, a inny ocenia podejmowane przez niego próby. Generator generuje coraz lepsze obrazy, natomiast dyskryminator uczy się lepiej je rozpoznawać, by móc odrzucać te nieprawdziwe. Po milionach odrzuceń system naśladujący nabiera wprawy w tworzeniu obrazów w stylu wybranego artysty, w efekcie czego dyskryminator nie będzie w stanie odróżnić ich od prawdziwych. Taki w pełni wytrenowany generatywny model może generować na żądanie obrazy, które będą niezwykle realistyczne.

CycleGAN

CycleGAN to technika, której ideą jest automatyczne uczenie modeli translacji obrazu na obraz bez konieczności posiadania sparowanych danych treningowych. Modele są szkolone w sposób nienadzorowany przy użyciu kolekcji obrazów z domeny źródłowej i docelowej, które nie muszą być w żaden sposób powiązane. Tworzony jest cykl, w którym dwa generatory współpracują ze sobą, aby przekształcić obraz w jedną stronę, a następnie to przekształcenie odwrócić. Można tutaj podać prosty przykład ilustrujący funkcjonowanie CycleGAN - zadanie polegające na zamianie zdjęcia konia w fotografię zebry i odwrotnie. Istotnie, jeśli generator AB najpierw przekształca konia w zebrę, to generator BA powinien przekształcić otrzymaną zebrę z powrotem w konia, a zrekonstruowany w taki sposób obraz powinien wyglądać identycznie, jak ten oryginalny. Takie podejście do problemu umożliwia generatorowi naukę tego, by nie generował on trywialnych zmian, a jedynie te, które składają się na krytyczne różnice między dwiema domenami. Inaczej nie byłaby możliwa powrotna konwersja obrazu. Metoda ta pozwala na wykorzystanie niesparowanych obrazów jako danych treningowych.

Wybór języka programowania

Projekt zrealizowany zostanie w języku Python, ze względu na dostępność licznych bibliotek ułatwiających przetwarzanie i analizę danych. Podczas pracy zespołowej wykorzystana

zostanie biblioteka Keras, która jest szeroko wykorzystywana podczas uczenia głębokiego i tworzenia sieci neuronowych. Rozważone zostanie także wykorzystanie TensorFlow, czyli opracowanej przez Google biblioteki open source do obliczeń numerycznych, często wykorzystywanej w zagadnieniach związanych z uczeniem maszynowym.

Podział zadań

Wykorzystanie platformy Kaggle zapewnia dostęp do odpowiednich zestawów danych, w związku z czym ten etap przygotowania projektu można uznać za gotowy. W wyniku zespołowej dyskusji, zaproponowano następujący podział zadania na etapy, który uwzględnia możliwość samodzielnego zapoznania się z tematem projektu i eksperymentów z implementacją:

1. We własnym zakresie - głębsze zbadanie tematu projektu, dostępnych poradników, materiałów edukacyjnych, publicznych fragmentów kodu, referencyjnych implementacji, oryginalnych artykułów opisujących mechanizmy działania sieci, etc.
2. Wspólna dyskusja i wyciągnięcie wniosków, podzielenie się najbardziej trafnymi materiałami, ocena dostępnych poradników i fragmentów kodu
3. Samodzielne eksperymenty z implementacją wybranych metod, wstępny projekt modeli i dobór parametrów
4. Wspólna dyskusja i wyciągnięcie wniosków, wybór najbardziej obiecujących metod / sposobów implementacji / wykorzystanych bibliotek
5. Następnie - podział obowiązków związanych z ostatecznym etapem przygotowania projektu na poszczególne osoby:
    a. Magdalena Falkowska - końcowa implementacja, przygotowanie finalnych wersji modeli i programu
    b. Dawid Alimowski - konfiguracja, trening modeli, dobór odpowiednich parametrów, generacja wynikowych obrazów
    c. Adam Bielecki - szczegółowe opracowanie wyników, porównanie zbiorów danych, podsumowanie prac nad projektem i wyciągnięcie wniosków

Uwagi

- Dokumentować indywidualne rozwiązania
- github

Źródła

- https://www.kaggle.com/c/gan-getting-started
- https://chmurowisko.pl/gan-ai-generujaca-rzeczywistosc
- https://atozofai.withgoogle.com/intl/pl/gans
- https://keras.io/
- https://www.tensorflow.org/api_docs/python/tf/all_symbols
- https://ichi.pro/pl/zamiana-plci-i-cyclegan-w-tensorflow-2-0-58960896505855
- https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/
- https://arxiv.org/pdf/1406.2661.pdf - GAN
- https://arxiv.org/pdf/1511.06434.pdf - DCGAN
- https://arxiv.org/pdf/1703.10593.pdf - CycleGAN