**ДЗ№0**

Найбільше мене цікавлять такі напрямки: Artificial Intelligence, Machine learning, Neural and Evolutionary Computing, Computer Vision. Іншими словам AI і Machine learning. Чому? По-перше це досить молоді галузі, в яких є багато перспектив, і в якій зараз дуже велика нестача спеціалістів. До того ж, Python тут найбільш розвинутий. Тобто під нього написано найбільше бібліотек, які стосуються штучного інтелекту і data science. І по суті це майже єдиний напрямок, де Python найкращий. Ну і найголовніше - мені це подобається. Завжди мріяв написати свій Skynet.

1) Насправді пан Андрій рекомендував не лізти в АІ, бо в нас там 0 досвіду. Але я з ним не згоден. Пайтон ком’юніті дуже дружнє та широке, в інтернеті є безліч хорошого матеріалу - багато туторіалів, статей, курсів і т.д. Тому я думаю, що як мінімум базові знання і готовий проект можна буде зробити за 8 тижнів.

2) Їх безліч. Дуже надихнувся проектом *deepfakes\_faceswap(*[*https://github.com/deepfakes/faceswap*](https://github.com/deepfakes/faceswap)*),* який в свій час був дуже популярним в соцмережах. Суть проекту - це підміна лиць людей у відео. Деякі такі підміни просто неможливо розрізнити від нормального відео, я вважаю що це неймовірний прогрес в АІ.

3) Як вже згадувалося вище: Artificial Intelligence, Machine learning, Computer Vision.

4) В даному циклі домашніх завдань я хочу дослідити тему розпізнавання предметів, зображень, облич з допомогою машинного навчання та нейронної мережі. Хочеться написати як свою мережу, яку можна було б тренувати, так і дослідити вже готові бібліотеки.

5) Думаю так. Звичайно, що теоретичну базу потрібно покращити, але я думаю, що тема не нереальна.

6) Поки що конкретно не можу сказати, який фінальний результат планується, але вже є кілька ідей, такі як аутентифікація по обличчю, пошук людини по обличчю в соцмережах (тут facebook має круте API) та покращення зображень.

7) Я вважаю, що всі в команді мають внести однаковий внесок (тавтологія).

8) Розібратися з наявними бібліотеками, ознайомитися з тим як працюють нейронні мережі та інші системи, які можна вчити. Думаю, що наявних знань в парсингу, роботі з файлами та інформацією, знання ООП, інших структур, та рівень написання коду на Python в мене достатній.

9) Поки що не досліджував, наскільки складно отримати facebook API, А так, ніби ніяких конкретних ресурсів не потрібно.

10) Перешкоди можуть виникнути на етапі реалізації власне програми. Просто дуже хотілося би зробити повноцінну програму під Windows, Linux чи Android, а з цим поки що не має ніякого досвіду. Було би круто зробити саме під Android, адже 2 з трьох запропонованих тем у 6) найкраще підходять під мобільний додаток. Але як відомо, програми під Андроїд зазвичай пишуться на Java або Kotlin. Звичайно існують бібліотеки (Kivy, PyJNIus), але я ще не досліджував можливості, які вони надають.

11) Звичайно. Я вже почав складати список матеріалів та думати над етапами. Проте я вважаю, що складати повноцінний план варто після узгодження команди.

12) Як показує досвід, найкраще розписати дедлайни аж до самого кінця, чітко розділити обов’язки. Дуже важливо оцінити реалістичність планів, і урізати/розширювати їх в залежності від того, як йде робота. Під час ознайомлення з ресурсами найкраще і найшвидше, щоби кожен обрав свої, а потім провів “гайд” по особисто вивченому матеріалу для інших учасників команди.

Я хотів би працювати в одній команді з Євгеном Панкевичом та Володимиром Цапівом. Наші зацікавленості і погляди на проект сходяться. Також ми вже почали разом планувати роботу та думати над конкретною темою проекту.

**Реферування статей.**

**1.** [**Building a Face Detection Model from Video using Deep Learning (Python Implementation)**](https://www.analyticsvidhya.com/blog/2018/12/introduction-face-detection-video-deep-learning-python/)

Автор - Faizan Shaikh - ентузіаст, який цікавиться машинним навчанням. З його [Linkedin](https://www.linkedin.com/in/faizankshaikh) профілю можна зробити висновок, що автор має освіту в [цьому індійському закладі](https://www.linkedin.com/school/pune-institute-of-computer-technology/?trk=public_profile_school_result-card_full-click). Також він працює в [Analytics Vidhya](https://www.linkedin.com/company/analytics-vidhya?trk=public_profile_experience-item_result-card_subtitle-click), де в нього є багато хороших статей, та власний курс по Deep Learning for Computer Vision. Ось його профіль в [Analytics Vidhya](https://www.analyticsvidhya.com/blog/author/jalfaizy/).

Я читав в [російському перекладі](https://habr.com/ru/company/netologyru/blog/434354/). Тим не менш, оригінал теж переглядав через кривий неробочий код в російському перекладі.

Стаття являє собою “супер” швидкий вступ до Computer Vision теми. По суті він розрахований на людей з нульовим рівнем знань у цій галузі. Тут повністю відсутня теорія та будь-які релевантні терміни. Напевно ідея в тому, щоби просто продемонструвати потенційні напрями реалізації технології, заохотити юних програмістів до вивчення цієї цікавої галузі.

Спочатку автор показує, які потенційні зручності вона може принести в повсякденне життя. Автор наголошує на тому, що технологія повинна використовуватися в етичних цілях. Очевидно, що автор натякає на глобальну систему розпізнавання облич у великих містах Китаю, яку влада може використовувати для тотального контролю населення. Як приклад автор показує, як в Китаї, в офісі для ідентифікації робітників використовується розпізнавання облич. Вражає.

Далі автор наводить найбільші сфери, де розпізнавання обличчя дуже корисне. (Соцмережі, безпека, підрахунок людей)

Далі починається практика. Варто сказати, що автор користується Ubuntu, тому не відомо, як даний код працюватиме на інших операційних системах. На щастя я користуюся дистрибутивом, заснованому на Ubuntu.

Найкорисніше, що можна дізнатися з цієї статі, це бібліотеки, які використовуються в прикладі. Базова бібліотека - OpenCV.

OpenCV - це крос-платформна бібліотека з відкритим кодом, головна мета якої - реалізація і створення програм з комп’ютерним баченням.

Також використовуються ще дві бібліотеки: dlib і face\_recognition. Сам автор зазначає, що dlib - найпростіша бібліотека з простим API. Справді, судячи по коментарях “диванних експертів” під статтею, dlib не годиться під серйозні проекти. Цікаво, що при першому встановленні dlib в мене крашнулась вся операційна система.

За кодом потрібно звертатися до оригіналу, бо в перекладеній статті відсутні всі відступи. Завдяки коментарям в коді, та поясненням автора зрозуміло, що яка стрічка коду виконує. По суті вся магія виконується в цій стрічці коду

[face\_locations = face\_recognition.face\_locations(rgb\_frame)]

І моя подальша мета буде заглянути під цей “капот”

Другий приклад є по суті тим самим, тільки тут відбувається робота з готовим відео, а не онлайн відео з веб-камери.

**Висновок.**

Мені однозначно сподобалася ця стаття. Попри те, що вона немає ніякої теоретичної інформації для розуміння, вона все ж таки ідеальна для того, щоб зрозуміти з чого почати, і які можливості та потенціал має ця галузь програмування.

**2)** [**A guide to Face Detection in Python**](https://towardsdatascience.com/a-guide-to-face-detection-in-python-3eab0f6b9fc1)

Автор - Maël Fabien. Судячи по його [Linkedin](https://fr.linkedin.com/in/mael-fabien), це серйозна людина, спеціаліст в Data Science, який має декілька вищих освіт, має широкий досвід роботи та розказує лекції. Також на [Medium](https://towardsdatascience.com/@mael.fabien) (сайт, де стаття і опублікована) в нього є багато інших цікавих статей

Ця стаття цікавіша, більша і детальніша, ніж попередня. Тут автор пробує пояснити певні базові концепти пов’язані з розпізнаванням лиць, такі як Adaboost, Viola–Jones framework, Haar Feature Selection, інтегральні зображення та інше. Тут вже з’являється невелике розуміння того, як ці алгоритми працюють.

Отже, які концепти важливо винести з цієї статті:

1) Haar Feature Selection. Так називають патерни, які характерні для людського обличчя. Наприклад, темна лінія очей, під якою знаходяться світлі щоки. Виразний ніс на прямій, яка проходить між очима, лінія губ. Оці особливі патерни в коді реалізуються так: Береться якийсь прямокутник (для різних патернів різні), на якому є світла і чорна частина. Очікується, що там де чорні пікселі прямокутника пікселі зображення будуть менш яскраві. і навпаки з білою частиною. І чим менша оця різниця між яскравістю пікселів прямокутника і зображення, тим більша імовірність того, що знайшлося обличчя.

Алгоритм, який цей принцип використовує називається Cascading Classifiers. Він використовується в бібліотеці OpenCV.

2) Adaboost. Це дуже “хитромудра” комбінація алгоритмів, яка допомагає прискорити пошук обличчя. Одні з таких алгоритмів враховує той факт, що не потрібно перевіряти на наявність облич все зображення кожного разу.

Як показує автор, цей алгоритм зменшує кількість можливих місць знаходження лінії очей для зображення 24х24 з 160000 до 200, з втратою точності лише на 5%.

3) Інтегроване зображення. Це зображення, кожен піксель якого дорівнює сумі значень усіх пікселів початкового зображення, чиї координати одночасно менші від координати даного пікселя. Ідея в тому, що замість того, щоби для кожного заміру значення пікселя звертатися до таблиці значень пікселів, можна з одного пікселя знаходити значення ближніх пікселів з допомогою простої рекурентної формули. Це дуже пришвидшує роботу програми.

Далі автор наводить два практичні приклади - для статичного зображення та відео з веб-камери. По суті код дуже схожий на той, що був в попередній статті за винятком того, що замість dlib він використовує вбудовані можливості OpenCV, а також підключає розпізнавання очей та роту. Варто також зазначити, що в цій статті автор не перетворює bgr (формат, в якому працює OpenCV) в rgb кольори, а робить зображення чорно білим.

Автор все ж використовує dlib. Як виявляється алгоритми цієї бібліотеки працюють по іншому. Називаються вони Histogram of Oriented Gradients (HOG).

Тут алгоритм не шукає патерни за значеннями яскравості пікселів в різних регіонах зображення, а будує таблицю градієнтів. Тобто вичислюється інформація про те, в якому напрямку в кожній точці найбільше змінюється яскравість, та наскільки. З цією інформацією набагато простіше шукати патерни обличчя.

Також пропонується реалізоване розпізнавання обличчя з веб-камери з допомогою dlib. По особистим відчуттям хочу сказати, що розпізнавання з допомогою Haar Cascade Classifiers (це назва алгоритму з OpenCV) працює набагато швидше, але менш стабільно і точно, ніж dlib. (Дуже цікаво, що автор зазначає в висновку абсолютно протилежне. Він стверджує, що HOG менш точний, але найкраще підходить для стрімінгових відео. Варто буде дослідити, чому так сталося)

Третій і останній спосіб розпізнавання зображень - нейрона мережа (Convolutional Neural Network (CNN)). Тут автор нажаль не вдається в деталі, та майже зразу пропонує практичну реалізацію.

Цікаво, що для реалізації CNN автор використовує dlib, хоча “дивані експерти” з попередньої статті говорили, що dlib погана, бо не підтримує CNN. Перша стаття вийшла майже на рік раніше від другої, тому можливо все змінилося за цей час.

**Висновок**

Чудова стаття, де автор крім хороших робочих прикладів пропонує хоча б на базовому рівні зрозуміти, як це все діло працює.

3) [How to Perform Face Detection with Deep Learning](https://machinelearningmastery.com/how-to-perform-face-detection-with-classical-and-deep-learning-methods-in-python-with-keras/)

Автор - Jason Brownee. Судячи по його [Linkedin](https://au.linkedin.com/in/jasonbrownlee), він викладає в [Swinburne University of Technology](https://au.linkedin.com/school/swinburne-university-of-technology/?trk=public_profile_topcard_school) та є активним учасником [Machine Learning Mastery](https://au.linkedin.com/company/machine-learning-mastery?trk=public_profile_topcard_current_company).

Чесно кажучи ця стаття мені сподобалася найбільше. По-перше, автор використовує цитати з книжок, що переконує у його знаннях. По-друге, читати цю статтю просто, написана вона зрозуміліше, ніж попередня.

Складається стаття з двох частин - класичне розпізнавання обличчя, та з допомогою deep-learning.

Під класичним розпізнаванням автор має на увазі Classifier Cascade. Як виявилося зі статті, цей алгоритм був придуманий 20 років тому. Про нього йшла мова в попередній статті, до того ж використовувалася та сама бібліотека для його реалізації (OpenCV), тому чогось реального нового я не дізнався з цієї частини. Проте деякі моменти стали більш ясними, що дало мені можливість виправити деякі помилки з рецензії на попередню статтю. На мою думку, автор тут краще пояснив основи реалізації алгоритму в бібліотеці.

Друга частина - deep-learning. В приклад автор наводить MTCNN (Multi-Task Cascaded Convolutional Neural Network). Цей алгоритм досить новий (2016 рік), і як можна здогадатися з назви, можна сказати, що це поєднання Classifier Cascade і CNN, тобто нейронної мережі. Алгоритм заведено розбивати на три етапи. Цитата зі статті:

“First the image is rescaled to a range of different sizes (called an image pyramid), then the first model (Proposal Network or P-Net) proposes candidate facial regions, the second model (Refine Network or R-Net) filters the bounding boxes, and the third model (Output Network or O-Net) proposes facial landmarks”

Теорії трохи мало, тому не дуже зрозуміло, як він працює. Зрештою алгоритм складний, тому пояснити його напевно нелегка задача.

Він має багато переваг перед класичними рішеннями. Найбільші - це а) набагато краща точність, б) можливість добре виявляти частини обличчя (очі, рот).

Для демонстрації використовується бібліотека mtcnn. В принципі робота з нею майже нічим не відрізняється від роботи з OpenCV за винятком інших назв класів та методів і відсутності реалізації збереження, чи виведення зображення на екран. Тому автор використовує matplotlib.pyplot.

В результаті ця цей алгоритм дає набагато точніші результати в порівнянні з Classifier Cascade, ще й ідентифікує очі та роти.

**Висновок**

Стаття хороша, лаконічна. Якісні приклади та хороші пояснення. Цікаво було почитати про MTCNN.