**1.1 Опис проблеми для вирішення**

Сьогодні сфера AI та Computer Vision розвивається дуже бурхливо, відбуваються нові великі дослідження та проекти. Проте степінь впровадження цих технологій дуже розчаровує. Якщо так подумати, то в голову приходять лише FaceID від Apple, і кілька кривих і лагучих AR проектів від Google, які не мають практичної користі. Кожен сучасний телефон має багато камер, кожен ноутбук має веб-камеру, і при цьому вони використовуються лише у найбанальніших цілях.

Кожного дня, ми сотні раз розблоковуємо свій телефон щоби перевірити пошту, чи месенджери. Існують кілька досліджень, згідно з якими в середньому людина заглядає в телефон 150-300 разів на день. Життя без телефону зробити неможливо, проте варто постаратися зробити його більш ефективним. Якщо припустити, що людина в середньому витрачає 3 секунди на розблокування, то виходить, що кожного дня ми витрачаємо від 8 до 15 хвилин просто на розблокування телефону. З ноутбуками ситуація повністю аналогічна.

Навіщо ми взагалі ставимо блокування на телефон, чи ноутбук? Питання риторичне - щоб приховати свою особисту інформацію, як повідомлення чи фотографії, від сторонніх людей. Але навіщо нам блокувати весь девайс, якщо ми хочемо приховати лише кілька програм? Було би набагато зручніше розблоковувати лише ті програми, дані в яких ми хочемо приховати. А ще зручніше було би це зробити з використанням ідентифікації по обличчю - швидше і простіше.

Варто замітити, що ідея ідентифікації по обличчю не нова, і навіть реалізована в деяких проектах. Найперше, що приходить до голови - FaceID, та розблокування по обличчю через фронтальну камеру в деяких Android оболонках. І все ж, їх ідея дещо інша від тої, що я пропоную. Вони лише зменшують час на розблокування, та роблять цей процес простішим. Крім того, зазвичай такий функціонал доступний на більш преміальних телефонах, а як відомо, більшість людей користуються дешевими Android смартфонами. Тим не менш функції поставити блокування на окрему програму із ідентифікацією по обличчю відсутня на більшості телефонах і операційних системах для ноутбуків. По суті, робочих, не лагучих і швидких аналогів просто не існує.

Варто зазначити, що розблокування по лицю в більшості Android смартфонів відбувається по алгоритмах, які розпізнають зображення. Іншими словами, безпечність такого блокування під великим питанням. Але існують алгоритми, які можуть відрізняти справжню тривимірну людину від двовимірного зображення, чи відео, як це наприклад продемонстровано [тут](https://www.youtube.com/watch?v=wr4rx0Spihs). Планується звичайно використовувати їх.

**1.2-1.5 Вибір та ознайомлення з API**

Існує декілька хороших бібліотека для роботи з зображенням та розпізнаванням обличчя. Найпотужніші: [OpenCV](https://pypi.org/project/opencv-python/), [dlib](https://pypi.org/project/dlib/), [mtcnn](https://pypi.org/project/mtcnn/). Всі вони надають різні алгоритми для розпізнавання та ідентифікації обличчя та функціонал для роботи з відео і зображеннями.

Для ознайомлення я обрав [mtcnn](https://pypi.org/project/mtcnn/). MTCNN - Multi-Task Cascaded Convolutional Neural Network - це дуже потужний алгоритм, який був придуманий досить недавно (2016 рік). Детальніше про принцип його роботи у 1.6.

Для використання модуля потрібно встановити [tensorflow](https://en.wikipedia.org/wiki/TensorFlow) - бібліотека від Google, мета якої створення і тренування нейронних мереж, а також сам модуль mtcnn.

pip3 install tensorflow

pip3 install mtcnn

**1.6-1.7 Короткий опис можливостей API.**

Приклад є у теці example.py. там є вся необхідна бібліотека для того, щоб зрозуміти, як він працює

Бібліотека ще хороша тим, що на ній є багато натренованих нейронних мереж. Наприклад [тут](https://github.com/davidsandberg/facenet) є мережа натренована на 3.3M фото, а точність 0.9965!

Існує багато [аналогів](https://github.com/open-face/mtcnn), які використовують той самий алгоритм. mtcnn має найпростіший API, тому його я і описував. Деякі аналоги, очевидно, мають потужніший функціонал.

Саме API дуже просте, і по суті містить лише один клас з одним публічним методом. При створенні екземпляру класу MTCNN важливо розуміти такі параметри:

weights\_file: str - файл, який містить усі ваги натренованої нейроної мережі.

Якщо не вказувати, то вибирається стандартний файл

min\_face\_size: int - мінімальний розмір лиця в пікселях. При збільшенні розміра покращиться продуктивність, але малі обличчя будуть ігноруватися

scale\_factor: float - фактор маштабування. Якщо дуже грубо, то чим більше це значення, тим уважніший алгоритм, і буде кращий шанс знаходження обличчя, але буде зменшуватися продуктивність. (Більше як 0.9 ставити не варто, бо відео дуже лагає, при 1 навіть не запускається).

Клас MTCNN містить метод ‘detect\_faces’, який один аргумент - img - зображення.

Повертає ‘detect\_faces’ список json формату, який виглядає так:

[ {

'box': [277, 90, 48, 63],

'keypoints':

{

'nose': (303, 131),

'mouth\_right': (313, 141),

'right\_eye': (314, 114),

'left\_eye': (291, 117),

'mouth\_left': (296, 143)

},

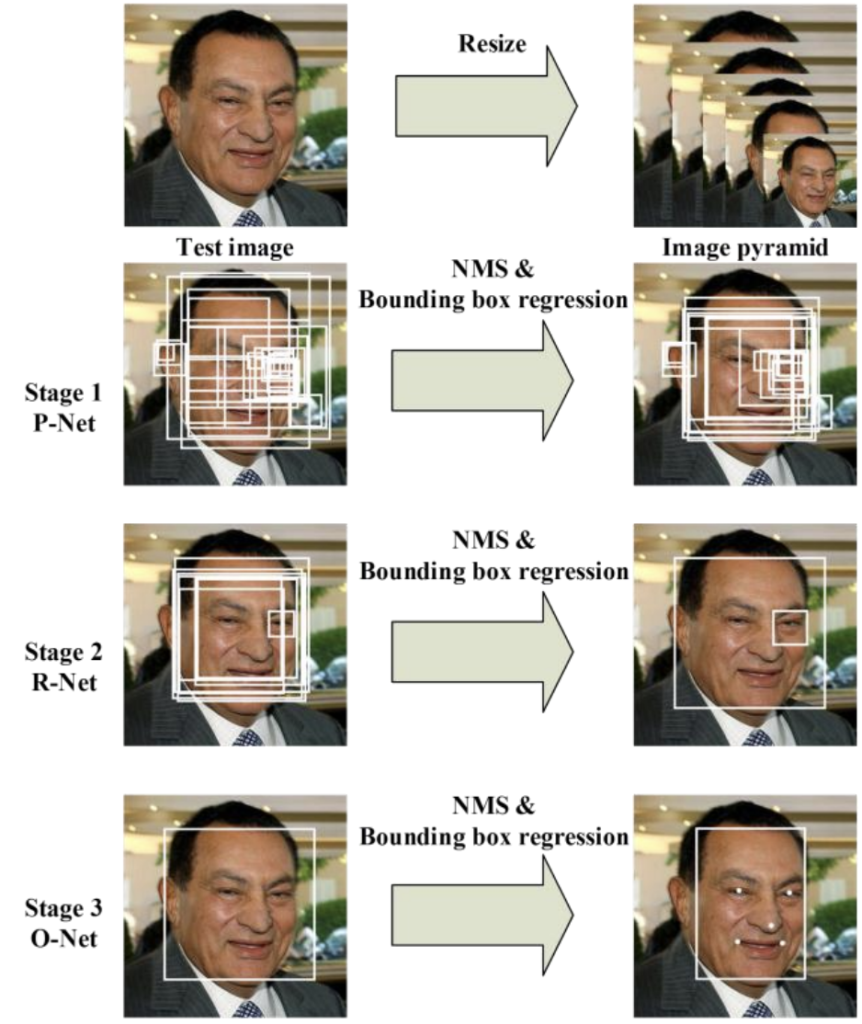
'confidence': 0.99851983785629272

}]

Кожен елемент репрезентує лице. ‘box’ - координати прямокутника, де знаходиться лице, ‘keypoints’ - словник з координатами носа, очей, лівого і правого краю рота.

Як працює алгоритм. Він називається ‘Multi-Task Cascaded Convolutional Neural Network’, бо в ньому є аж 3 нейроні мережі. Працює він в 4 етапи:

1. Зображення масштабується в багато зображень з різними розмірами
2. P-Net (Proposal Network) мережа швидко проходиться по усім зображенням, і вибирає кілька кандидатів, які зустрічаються на усіх змасштабованих зображенням
3. R-Net (Refine Network) мережа більш строга. Вона відсіює усіх поганих кандидатів на знаходить усі справжні обличчя
4. O-Net (Output Network) мережа знаходить усі частини тіла (очі, ніс, рот)

Зображення знизу демонструє, як це відбуважться: 

**1.8 The Systems Analyst and Information Systems Development.**

Спонсори проекту: Павло Гілей, Євген Панкевич, Володимир Цапів

Бізнес потреба: Сенс цього проекту - покращити і спростити UX для користувача, давши йому вибір того, що він хоче заблокувати, та надати йому дуже швидку і просту ідентифікацію для доступу до прихованого матеріалу/програмного забезпечення.

Бізнес вимоги:

* Модифікувати наявні алгоритми для аутентифікації з допомогою обличчя для кращої продуктивності та безпеки
* Створити робочу Android програму, яка дозволятиме користувачу:  
  - Вибирати програми, доступ до яких користувач хоче обмежити.  
  - Встановити запасний пароль або спеціальний рисунок для доступу до програми, якщо відсутній доступ до обличчя.  
  - Можливість вибирати час, через який ідентифікація буде знову вимагатися  
  - Можливість миттєво заблокувати всі програми  
  - Можливість кастимізувати зовнішній вигляд програми  
  - Включити функцію, яка фотографуватиме іншу людину, яка пробує доступитися до прихованої інформації
* Портувати програму на Windows, Linux

Бізнес вигоди:

Користувачі отримають безплатну, open-source програму, яка не буде містити реклами, програма буде практична, і буде практичною. Тому користувачі будуть отримувати задоволення від користування нею.