Part 1

У першій частині цієї лабораторної роботи спробуємо описати, як саме ми створювали класифікатори, із якими труднощами зіткнулися й наведемо порівняльну табличку класифікаторів і дескрипторів для зображень і зробимо по ній певні висновки.

Отож розпочнімо із нашої вибірки. На основі попередньої лабораторної роботи ми створили новий датасет зображень, який містив зображення із машинкою BMW X6 (тут і далі класифікується класифікатором як 1), кораблем барк (тут і далі класифікується класифікатором як 2) і зображення без корабля й машинки (тут і далі класифікується класифікатором як 0). Усього вийшло 269 зображень. (Спойлер! Далеко не всі із них ми будемо використовувати для наших класифікаторів, але про це згодом.)

На цьому етапі варто відзначити, що ми використовуємо результати нашого дескриптора для кожного зображення, як основні фічі для одного трейн екзамплу. І звісно ж, для адекватного порівняння дескрипторів і класифікаторів ми використовуємо один датасет зображень, який ми описали вище. Проте тут шляхи кожного члена команди розходяться й він, маючи датасет і свій дескриптор, старається обробити ці фічі таким чином, щоб максимізувати точність кожного із класифікаторів.

Тепер нехай кожен член команди пояснить свій підхід для обробки даних, які він отримує від дескриптора (обробка фіч).

- Я (Володимир Возняк) розглядав дескриптор ORB. Основною відмінністю цього дескриптора від інших є те, що при його ініціалізації можна задавати к-сть фіч, які буде шукати дескриптор на фото. Для цієї лабораторної я вирішив взяти nfeatures = 500. У такому разі для кожного зображення дескриптор ідеально мав би знаходити матрицю значень розмірністю [500x32], проте для деяких фото він не знаходить фіч взагалі, або знаходить менше 500 (скоріш за все ці фото є дуже затемненими або розмитими, оскільки, як ми уже дізналися на попередній лабораторній, ORB є дуже чутливим до таких зображень). Тому я вирішив використовувати лише ті зображення із тестової вибірки, на яких дескриптор знаходить рівно 500 фіч (таких зображень виявилося 221). Тобто на виході для кожного зображення із датасету ми отримуємо матрицю значень [500x32], кожне значення якої ми й будемо використовувати, як фічу для одного трейн екзампла (у

нашому випадку це одне зображення). Тобто на виході я отримав матрицю X розмірністю [221x16000], де 221 – к-сть трейн екзамплів (к-сть зображень), а 16000 – к-сть фіч. Для такого варіанту (коли к-сть фіч набагато більша за к-сть трейн езкамплів) ідеально підходять Logistic Regression та SVM with linear and polynomial kernels. Саме тому першим ділом я вирішив реалізувати саме ці класифікатори (детальніше про них у моєму Джупітерському файлі на Гіті, посилання на який є вкінці файлу). Щоб використати інші класифікатори (детальніше про них усіх у Джупітерському файлі, або в табличці) я застосував Principal Component Analysis (PCA) із n_components = 100, або так зване Dimensionality Reduction (зниження розмірності) і перетворив вхідну матрицю X розмірністю [221x16000] на матрицю [221x100]. Звісно, що при такому підході втрачається дуже багато інформації, проте класифікатори працюють набагато краще, ніж якщо взагалі не застосувати PCA. Щодо нейронки, то тут підхід був аналогічним до підходу Андрія (про нього трішки нижче). Варто також відзначити, що я пробував підхід Андрія для всіх вище наведених класифікаторів, але отримував набагато меншу точність обчислення (лише для нейронки - вищу), тому я залишив свої спроби, як найкращі, які максимізують точність роботи кожного класифікатора (а як говорилося вище, це саме те, чого ми прагнемо).

- Я (Аблець Андрій) розглядав дескриптор KAZE. На відміну від ORB, KAZE повертав матрицю значень з довільною кількістю рядків, в ньому лише можна було задавати кількість стовпців ступенів двійки. Тому треба було вирішувати якось цю проблему, оскільки для використання моделі потрібна однакова кількість параметрів. Було вирішено використати метод головних компонент (PCA), з кожного рядка довільної розмірності створити розмірність один. В результаті для кожного трейн екзамплу вийшло по 64 фічі, тобто на виході я отримав матрицю X розмірністю [262x64], де 262 – кількість трейн зображень, а 64 – кількість фіч. Як можна побачити, KAZE менш прискіпливий за ORB та викинув лише 7 зображень (ORB викинув 48). Це і не дивно, оскільки ORB дуже сприятливий до розмитих фотографій, а розмитих фотографій у нас достатньо. Одразу я спробував реалізувати нейронну мережу без використання дескрипторів і порівняти її з нейронною мережею на основі дескрипторів. Різниця в точності вийшла доволі значна: 0.87 та 0.67 на користь дескрипторів. Я гадаю, що проблема у маленькій роздільній якості (150x150), до якої довелося зменшити наші

фотографії заради швидкодії, але все одно програма працювала доволі довго. В той же час при роботі з дескрипторами використовувалася роздільність 1280x720. Ще одною проблемою було те, що в нас була доволі маленька вибірка (лише 269 фотографій). Цю проблему можна вирішити доповненням даних методом отримання даних з існуючої трейн вибірки з використанням випадкових перетворень. Але навіть з такими хитрощами вона буде знаходиться далеко від мережі з дескрипторами, принаймні по часу – так точно, оскільки звичайна працювала десь 25 хвилин, а дескрипторна не більше хвилини. Після цього я спробував використати декілька інших моделей, але все одно нейронна мережа була на першому місці, проте по часу була на останньому. Найближче до неї підібрався SVM with linear kernel, в інших точність була нижча. Щодо порівняння точності різних класифікаторів у різних дескрипторів, то можна сказати, що KAZE справляється значно краще за ORB, час роботи у KAZE теж вище, окрім SVM with linear kernel та нейронної мережі, це можна побачити у наведеній нижче таблиці. Це, в принципі, і не дивно, беручи до уваги наше порівняння цих класифікаторів у попередній практичній роботі.

Тепер трішки пояснимо щодо основної метрики, яку ми будемо використовувати для порівняння класифікаторів та дескрипторів. У завданні лабораторної роботи вказувалось “прогнати отримане комбо з класифікатора і дескриптора на тестовій вибірці збираючи як позитивні результати так і помилки першого і другого роду”. Це зробити максимально просто за допомогою confusion matrix. Ось, наприклад, дві confusion matrix для одного із train_test_split для Logistic Regression (зліва – Возняка Володимир, справа – Аблеця Андрія).

```
print(metrics.confusion_matrix(y_test, y_pred))
```

```
[[ 0  1  4]
 [ 0 19  5]
 [ 0  2 25]]
```

```
print(metrics.confusion_matrix(y_test, y_pred))
```

```
[[ 5  2  0]
 [ 2 24  1]
 [ 2  1 29]]
```

Із неї досить чітко видно кількість правильно спрогнозованих фото та помилки першого й другого роду. Наприклад, розглянемо confusion matrix Возняка Володимира. Класифікатор правильно знаходить 19 машин на фото та 25 кораблів, проте замість того, щоб не знайти нічого, він 4 рази знаходить кораблі й 1 раз машинку, а замість того, щоб знайти машинку, він 5 разів знаходить корабель і тд.

Так, confusion matrix – дуже хороша метрика, проте для нашого датасету ідеально підійде метрика точності, обчислена за допомогою cross validation, оскільки в нас майже порівну зображень із машинкою й кораблем і десь вдвічі менше зображень без машинки й корабля. А в такому разі є досить доцільним користуватися саме такою метрикою. Тому будемо користуватися точністю по cross validation, як основною при порівнянні дескрипторів і класифікаторів.

Прийшов час створити табличку для дескрипторів і класифікаторів із точністю по cross validation та часу роботи кожного класифікатора, із якої зразу буде видно, який із класифікаторів та дескрипторів кращий.

	ORB		KAZE	
	Accuracy	Time	Accuracy	Time
Logistic Regression	74%	12.5s	83%	128ms
SVM with linear kernel	71%	12.5s	84%	237ms
SVM with polynomial kernel	71%	12.8s	77%	73ms
SVM with Gaussian kernel	72%	99ms	79%	81ms
Random Forest	71%	2.42s	85%	5.2s
Decision Trees	60%	119ms	83%	108ms
Neural Network	65%	2s	87%	10.7s

Висновки

Наведемо деякі висновки стосовно цієї таблички й порівнянні дескрипторів і класифікаторів у форматі тез.

- Дескриптор KAZE справляється із цим завданням набагато краще за дескриптор ORB (у середньому на 10%, а то й більше, для кожного із класифікаторів), що впринципі очевидно, вважаючи наше порівняння цих дескрипторів у попередній лабці.
- Така різниця може бути викликана тим, що ми застосовували різні підходи, щоб максимізувати відсоток точності для кожного із класифікаторів (можливо, Андрій впорався із своїм завданням трішки краще ☺). Хоча, на нашу думку, ця різниця викликана перш за все через те, що KAZE знаходить на деяких фото до 20000 фіч, у той час як ORB лише фіксовану кількість – 500. І навіть із застосуванням PCA і зменшенням розмірності із, наприклад 20000, до 1 KAZE містить більше інформації про фічі зображення, ніж ORB без застосування PCA.

- Трішки щодо часу. Як ми бачимо класифікатори без застосування PCA раняться досить довго в порівнянні із іншими класифікаторами. І в середньому класифікатори із KAZE працюють швидше (та це викликано виключно нашими особливостями реалізації програм).
- На останок наведемо найкращий класифікатор взагалі та для кожного із дескрипторів. Як і говорилося раніше, перша думка Володимира про те, що Logistic Regression ідеально тут впишеться для ORB була вірною й дійсно саме вона видає найбільший відсоток точності для даного класифікатору. Щодо найкращого класифікатору для KAZE та взагалі, то тут переможець – нейронна мережа. Дійсно, дану нейронку в ідеалі можна натренувати просто до космічного рівня точності, збільшивши трейн вибірку й зробивши ще деякі махінації. Проте навіть для такої невеличкої трейн вибірки вона вже видає результат у 87% точності по cross validation, що ми вважаємо супер-круто.

Part 2

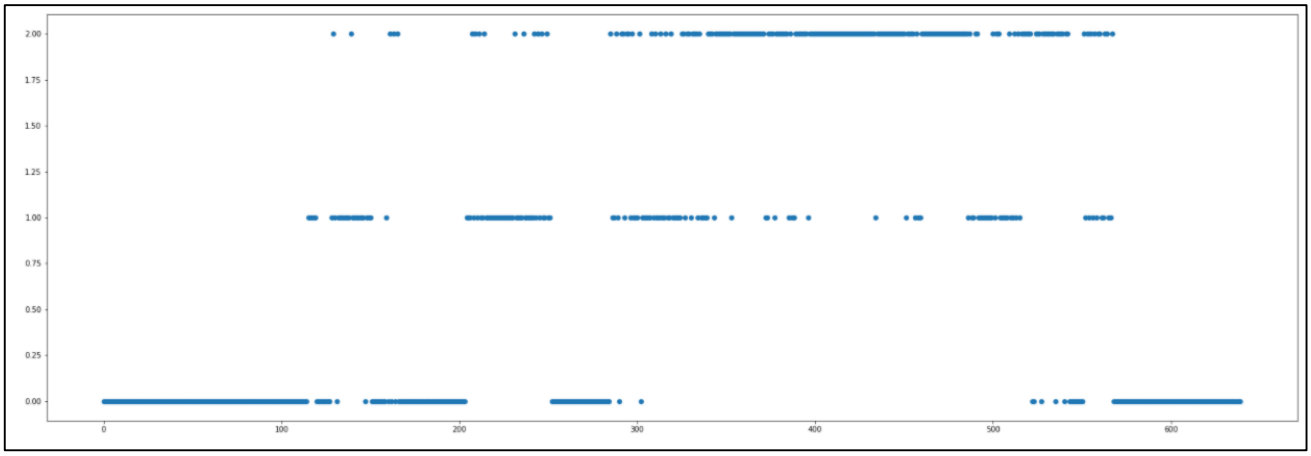
У другій частині цієї лабораторної роботи ми спробували погратися із відео, на якому спочатку появлялася машинка, а згодом корабель. Окрім того, щоб розбити це відео на окремі кадри й ці кадри використовувати аналогічно тому, як ми робили в тестуванні класифікаторів із зображеннями, нічого більше в голову не прийшло, тож ми реалізували саме цей варіант.

Як і раніше, кожен член команди намагався максимізувати точність своєї окремо виконаної роботи, тому використовував класифікатор, який йому найбільше припав до душі, базуючись на результатах попередньої частини.

Перейдемо до описів кожного із членів команди (показ результатів і деякі коментарі).

Я (Володимир Возняк) перш за все вирішив обробляти відео за допомогою Linear Regression, оскільки саме вона давала найбільшу точність (близько 74% по cross_validation). Проте при такому підході, як і раніше, я мусив використовувати зображення, на яких дескриптор у точності знаходить 500 фіч. У такому разі майже всі кадри, які не містили машинки або корабля, цю перевірку не проходили, тобто до кінцевих тестових фото дійшли лише ті кадри, які містять або машинку, або корабель. Звісно, я згідний, що такий підхід трішки не коректний, проте моя програма видає, для яких саме кадрів вона не може знайти 500 фіч, а

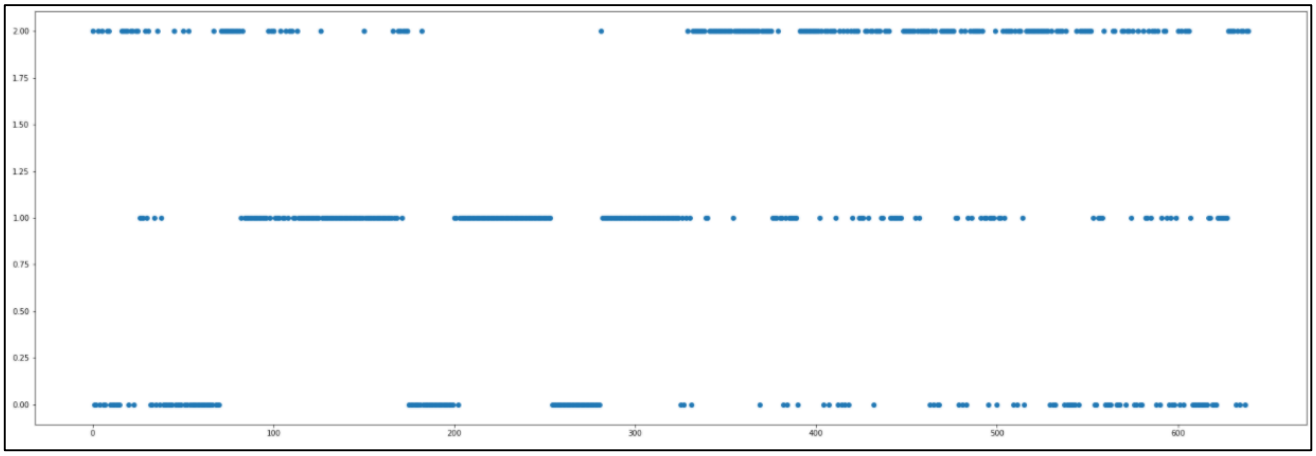
ми цілком вірогідно можемо інтерпретувати це, як нулі (тобто як те, що на цьому кадрі немає ні корабля, ні машинки). Тобто ми вже знаємо, на яких кадрах немає ні машинки, ні корабля. Тепер глянемо на динаміку, яку пропонує нам Logistic Regression для вхідного відео.



По вигляду графіку можна зробити висновок, що Logistic Regression більш-менш справляється із своїм завданням. На початку йдуть 0, що означає, що на кадрах немає ні машинки, ні корабля, тоді починаються 1, що означає, що класифікатор оприділяє на кадрах машинку, із подвійним переривом на 0, де я закриваю рукою камеру, а ближче до середини починають переважати 2, що означає, що класифікатор оприділяє на цих кадрах корабель, і вкінці ми знову спостерігаємо 0, оскільки Андрій відводить камеру від корабля. Нагадаю, що на кадрах, які не включають у себе ні корабель, ні машинку, дескриптор не може знайти достатню к-сть фіч, тому ми їх просто не беремо до уваги, а вважаємо, що на їхньому місці має бути 0, тобто, що там немає ні машинки, ні корабля (знайдення не достатньої к-сті фіч дескриптором у більшості випадків означає, що зображення занадто затемнене або розмите, що грає дуже валику роль для дескриптора ORB, тому цілком логічно інтерпретувати ці кадри, як 0 і я “штучно” добавив ці 0 до результату класифікатора).

Все ж таки попередні викладки були трішки притягнутими за вуха, тож я вирішив ще затестити поведінку нейронки при обробці відео. Так, нейронка давала досить малий відсоток точності в порівнянні із іншими класифікаторами, проте вона єдина брала до уваги фото, для яких дескриптор знаходить менше 500 фіч. Проте навіть у такому випадку знаходяться фото, для яких дескриптор взагалі не знаходить жодної фічі (це кадри, де я двічі закриваю камеру рукою, коли знімаю машинку, що досить логічно, адже вони получаються дуже затемненими). Аналогічно до попереднього випадку, інтерпретуватимемо ці значення як 0 і

добавимо їх до класифікатору (тут уже це не настільки притягнуто за вуха, оскільки ми не враховуємо лише фото, де дескриптор взагалі не знаходить жодної фічі й цілком природно вважати ці кадри рівними 0). Тож глянемо на результати роботи нейронки.



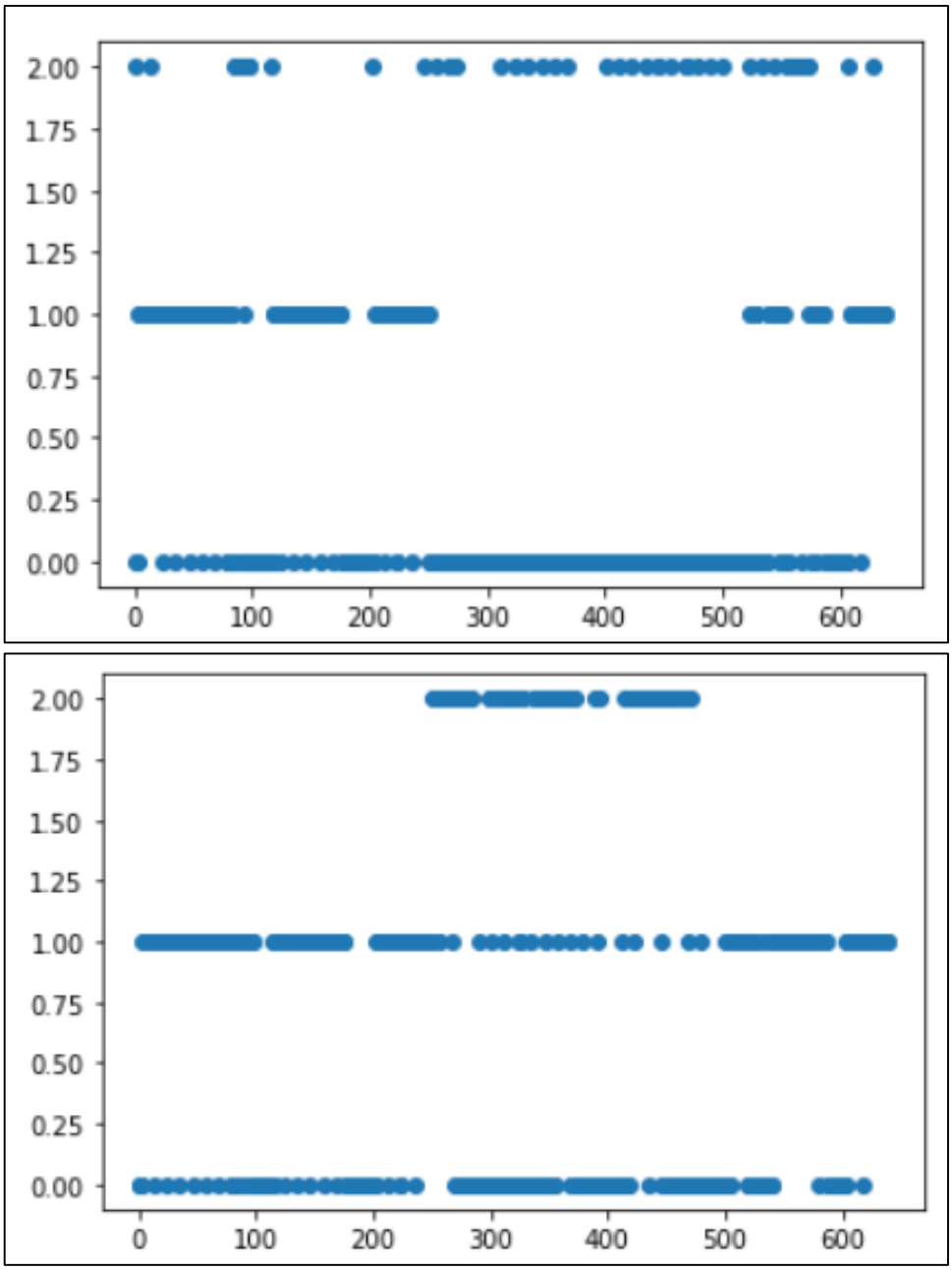
Ми бачимо, що впринципі динаміку відео нейронка розпізнає вірно. Спочатку переважають 0, що означає кадри без машинки й корабля, потім ідуть 1, що означає кадри, на яких є машинка із переривами на 0 (моменти, коли я закриваю камеру рукою) і потім переважають 2, що означає кадри, де є корабель. Чомусь саме корабель нейронка розпізнає набагато гірше за машинку. Можливо, через фон, на якому розташований корабель, а можливо, через те, що Андрій перевертав цей ж корабель та знімав його зверху, але ось чому там з'являються 1, пояснити ніяк не можна (скоріш за все просто через замалу трейн вибірку). Вкінці ж динаміки починають переважати 0, що цілком правильно, бо Андрій відводить камеру від корабля вбік.

Отже, ось так працює класифікатор при обробці відео. На мою думку, все ж нейронка краще показує саме динаміку об'єктів на кадрах, проте дуже часто помиляється для окремо взятих кадрів. Я вважаю, що із збільшенням трейн вибірки, нейронка почне працювати краще й збільшить свою точність.

Передаю слово Андрію ☺

Я (Аблець Андрій), оскільки в мене не було такої проблеми як у Володимира, KAZE знаходив багато кадрів, що не включають ні корабель, ні машину, вирішив порівняти роботу з відео нейронної мережі та логістичної регресії. Оскільки ми все-таки вирішили використовувати у тренуванні моделей дескриптори (так як було у початковому завданні), я спочатку був впевнений, що нейронна мережа буде значно попереду у порівнянні точностей, але все

виявилося не так як я очікував. Спочатку продемонструю розподіл фреймів по класам (зверху нейронна мережа, знизу регресія) .



Оскільки 0 – це інші предмети, 1 – машинка, 2 – кораблик, то можна вже на цьому етапі прийти до висновку, що логістична регресія справилася не гірше. Вони обидві не дуже гарно працюють: нейронна мережа краще відображає машинку, а регресія краще відображає корабель. Але, маючи досвід класифікації фотографій, я все ж таки думав, що буде значна різниця у точності. Проблема в тому, що ця нейронна мережа дуже залежить від початкового моменту ініціалізації, а він випадковий. І оскільки в нас дуже маленька трейн вибірка, ми

отримуємо такі результати. Але в середньому нейронна мережа все-таки буде краще працювати. Та все-таки краще за все подивитися представлені нами відео зі склеюванням класифікованих зображень з вотермаркою до якого класу вони належать. Щодо порівняння KAZE і ORB, то було досі очевидно, що KAZE буде краще, це ми ще перевірили у попередній лабораторні. А зараз, отримуючи такий результат, ми впевнюємося, що все зроблено вірно.

Висновки

- Варто відзначити, що в якості основного класифікатору для обох дескрипторів, було обрано нейронну мережу.
- Нейронна мережа в комбінації із дескриптором KAZE працює набагато краще, проте й нейронка із ORB доволі непогано розпізнає основну динаміку відео.
- Усі викладки, які наводилися вище, представлені, як результат у формі відео із текстовим оверлеєм для кожного кадру вхідного відео. Їх можна знайти за посиланням на Гіт кожного із учасників команди, які знаходяться нижче.

Дякую за увагу!

Made by IASA_HEROES

Посилання на програми учасників команди:

- Дескриптор **ORB** (Возняк Володимир, КА-71): https://github.com/VolodymyrVozniak/CVPR (lab_3)

- Дескриптор **KAZE** (Аблець Андрій, КА-71): https://github.com/AndreiAblets/CVPR (lab3)