ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ

Факултет по телекомуникации

Специалност: Телекомуникации

Курсов проект

по „Основи на Аудио и Видео Технологиите“

Тема:

Основни характеристики на дълбоко машинно обучение за компютърно зрение

Студенти:

Михаил Александров Александров	Фак. №: 111217168	Група: 51
Живко Руменов Йорданов	Фак. №: 111217169	Група: 51

Преподавател: **гл. ас. д-р. Николай Нешов**

Дата: ..... Подпис: .....

София, 2021

Съдържание

Катедра: Радиокомуникации и видеотехнологии
Дата на задаване: 28.09.2020
Краен срок на предаване: 23.01.2020
Образователна степен: бакалавър, специалност: ТК

1. ЗАДАНИЕ

ЗА КУРСОВ ПРОЕКТ

ПО „ОСНОВИ НА ВИДЕО И АУДИО ТЕХНОЛОГИИТЕ“

на студентите:
Живко Руменов Йорданов, гр. 51, Фак. номер: 111217169
Михаил Александров Александров, гр. 51, Фак. номер: 111217168

Тема: 35. Основни характеристики на дълбоко машинно обучение за компютърно зрение

Съдържание на курсовата работа *(8-10 стр. при изпълнение от един студент и 15 стр. при изпълнение от двама студенти, Размер на шрифта - 12, междуредие - 1,5)*

1. Заглавна страница/ задание
2. Увод
3. Анализ на текущото състояние на проблема по литературни данни *(цитиране на източниците в края на проекта)*
4. Описание на проблема *(матет. формули, фигури, таблици и др.)*
5. Реализация *(описание на алгоритми, хардуер, програмен код на симулации, блокови схеми и т.н.)*
6. Примери за приложение
7. Използвана литература *(цитиране на източниците)*

Студент: .....
(Живко Йорданов)

Студент: .....
(Михаил Александров)

Научен ръководител: .....
(гл. ас д-р инж. Н Нешов)

2. Увод

Технологията за компютърно зрение е една от най-обещаващите области за научни изследвания в рамките на изкуственият интелект и компютърните науки, и предлага огромни предимства за бизнеса в нашето съвремие. В сърцевината си, областта на компютърното зрение се фокусира върху проектиране на компютърни системи, които притежават способността да улавят, да разбират, и да интерпретират важна визуална информация, която се съдържа в изображенията и видео данните. След това компютърните визуални системи превеждат тези данни, използвайки контекстуални знания, предоставени от хора, използвани за задвижване на вземането на решения. Превръщането на едно изображение от необработен вид в концепция от по-високо ниво, така че хората или компютрите да могат да интерпретират и действат по него, е основната цел на технологията за компютърното виждане. Трябва обаче, да се направи важно разграничение между компютърното зрение и обработката на изображения. Обработката на изображения е науката за извършване на промени на един образ, така че да се произведе нов образ с определени (засилени) характеристики. Тези промени може да включват – увеличаването на резолюцията, нормализиране на яркостта и контраста, изрязване, замъгляване, или някаква друга дигитална трансформация, която би била необходима за конкретната цел. Дигиталната обработка на изображения не взема в предвид действителното съдържание на изображението - това е просто поредица от механични трансформации, предприети да променят изображението за някаква определена цел.

Системите за човешкото зрение имат огромното предимство да бъдат информирани за знания за цял живот, които помагат да се контекстуализират данните в нашето зрително поле. Нашите очни ябълки улавят визуалната информация например - образа на една котка, и нашият предишен опит интерпретира това събиране на отразена светлина и се свързва с понятието за котка. Сложността на нашата система за визуално възприятие и тясната ѝ връзка с нашата памет и по-високи способности за разсъждение дава на тези визуални данни необходимият контекст, за да осигури качество в ежедневните активности. Тези човешки

способности, докато са недостъпни за компютрите, могат да бъдат имитирани ефективно чрез алгоритми за машинно обучение. Но както се оказва, обучаването на машини, които да имитират тази основна човешка функция, която е гордо демонстрирана от 5 годишни деца по цял свят, е изключително трудно. Решаването на този проблем непрекъснато заема най-ярките умове в изследванията на изкуственият интелект.

3. Анализ на текущото състояние на проблема по литературни данни

3.1 Какво е машинно обучение

Машинното обучение е знанието, че компютрите се учат и действат като хора. Машинното обучение е компютърната парадигма, която води до растежа на "големите данни" и изкуствения интелект. То се основава на развитието на невронни мрежи и дълбокото обучение. Обикновено това се описано като имитация на начина, по който хората се учат, но това е малко погрешно определение. Машинното обучение всъщност се отнася до статистически анализ и итеративно обучение. Има много различни видове алгоритми за машинно обучение. Всеки ден се публикуват стотици. Те обикновено са групирани стил на учене (контролирано обучение, неконтролирано учене, обучение с полу-контролиран достъп) или по-късно споразумения във форма или функция (напр. класификация, регресия, дърво на решенията, групиране, задълбочено обучение и др.). Независимо от стила или функцията на обучение, всички комбинации се състоят от следното:

- представителство (набор от класификатори или езика, който компютърът разбира)
- оценка (известен също като функция за обективно / отчитане)
- оптимизация (методът за търсене, често най-добрият класификатор за оценяване, се използват както методите за оптимизация,

Основната цел на алгоритмите за машинно обучение е да се обобщава допълнително, т.е. да се интерпретират успешно данни, които никога преди не са били представени. Има различни начини да оставим машините да се учат. От използването на обикновени дървета за вземане на решения до групиране на слоеве от изкуствени невронни мрежи (последното е дало възможност за дълбоко учене). Докато фокусът често е върху избора на най-добрия алгоритъм за обучение, изследователите са открили, че някои от най-интересните въпроси не са отговорени от наличните ресурси. Важен момент е, че машинното обучение не е просто, а дори и автоматизация. Ако мислите така, вие със сигурност пропускате ценните прозрения и възможности, които машините могат да ви предложат.

Машините, които учат са полезни, тъй като с цялата си обработваща мощност те могат по-бързо да намират модели в големи данни. Иначе тези части биха били пропуснати от хората. Машинното обучение е инструмент, който може да се използва за подобряване на решаването на проблеми. Той може да направи информирани заключения в по-широк спектър от проблеми като диагностициране на болести и съставяне на решения за борба с глобалното изменение на климата. Срещат се и обаче предизвикателства и ограничения при машинното обучение. Една от най-честите грешки при начинаещите е успешното тестване на данните за обучение с илюзията, че сте успешни. Част от събирането на данни не казва нищо за целия набор от данни. Когато алгоритъмът за обучение (т.е. ученикът) не работи, понякога той успее, като доставя повече данни на машината. Това обаче може да доведе до проблеми със мащабируемостта, където имаме повече данни, отколкото времето за учене. Машинното обучение не е самоцел. Освен това не е полезно да се използва като общо решение. Веднъж обучен, алгоритъмът за машинно обучение е в състояние да сортира чисто нови входове през мрежата с голяма скорост и точност в реално време. Това го прави основна технология за компютърно зрение, гласово разпознаване, езикова обработка, както и научноизследователски проекти.

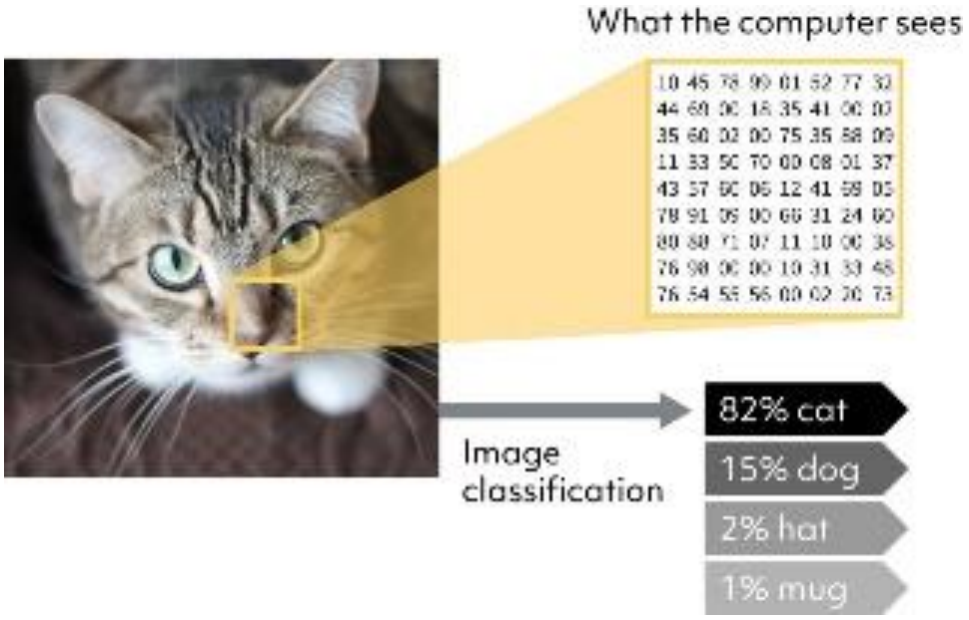
3.2 Какво е дълбоко обучение

Дълбокото обучение или йерархичното обучение е подмножество на машинното обучение в изкуственият интелект, който може да имитира функцията за обработка на данни на човешкия мозък и да създава подобни модели, които мозъкът използва за вземане на решения. Противно на алгоритмите, базирани на задачи, системите за дълбоко обучение се учат от представянето на данни - те могат да се учат от неструктурирани или немаркирани данни. Архитектурите за дълбоко обучение като дълбоките невронни мрежи, мрежи от вероятности(бейсови мрежи), повтарящи се невронни мрежи, и конволюционните невронни мрежи са намерили приложения в областта на компютърното зрение, аудио/говорното разпознаване, машинният превод, филтрирането в социалните мрежи, биоинформатиката, дизайнът за лекарства и много други.

3.3 Какво представляват невронните мрежи

Невронните мрежи са съставени от набор от алгоритми, които са моделирани спрямо структурата на човешкият мозък. Тези алгоритми могат да интерпретират сензорни данни чрез машинно възприятие и надписвайки или групирайки необработените данни. Те са предназначени да разпознават числови модели, които се съдържат във вектори, в които са всички данни от реалния свят (изображения, звуци, текстове, времеви редове и т.н.) трябва да бъдат преведени. В основата си, основната задача на невронните мрежи е да групира и класифицира необработените данни - те групират немаркираните данни въз основа на приликите, които са намерени във входните данни и след това се класифицират данните въз основа на етикетираният набор от данни за обучение. Невронните мрежи могат автоматично да се адаптират към променящият се вход. Така че, не е необходимо наново да проектираме изходните критерии всеки път, когато входът се промени, за да генерира възможно най-добрия резултат.

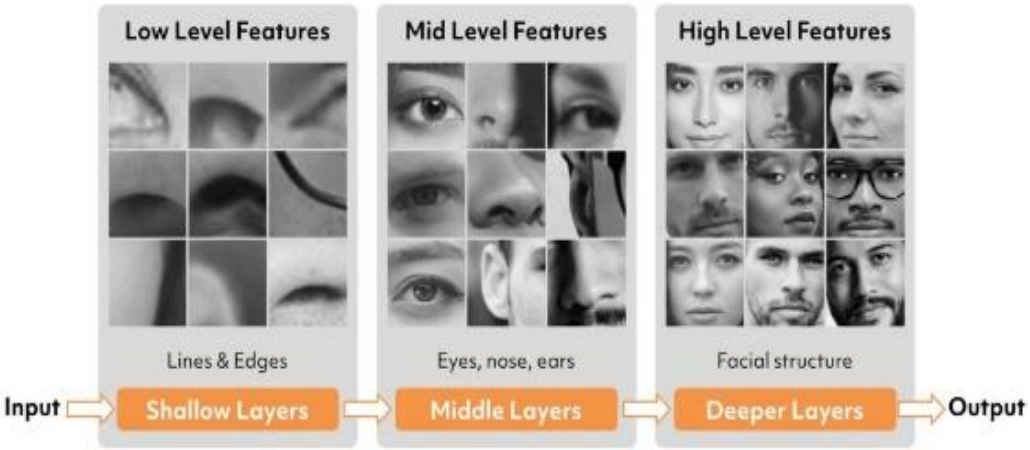
Невронните мрежи понастоящем са най-популярният начин да се направи Deep Learning, но съществуват и други начини за постигане на машинно обучение. Дълбокото обучение в компютърното зрение стана възможно чрез изобилието от данни за изображения в съвременния свят, плюс намаляването на разходите за изчислителната мощност, необходима за обработването му. Големи мащабни комплекти от изображения като ImageNet, CityScapes и CIFAR10 събраха милиони изображения с точно обозначени функции за алгоритми за дълбоко обучение. Привидно за една нощ, изпълнението на алгоритмите за дълбоко обучение надмина тридесет години работа по ръчни детектори на функции. Допълването на достатъчен брой добре обозначени изображения към дълбока визуална система, която дава възможност да се разберат точните нюанси на ниво пиксел и определя отделните компоненти на по-голямото изображение. Тя автоматично ще научи къде са ръбовете, и как особени комбинации от ръбове, които се различават по цвят и контраст един от друг и от фона се комбинират, за да образуват определени функции. Изображението на Фиг. 1 показва как една система за дълбоко обучение може да идентифицира характеристиките на котка.



Фиг. 1

3.4 Какво са конволюционни слоеве

Конволюционните слоеве от по-висок ред на невронната мрежа ще започнат да разбират, че ако на изображението има 4 крака, глава, опашка и тяло, следователно въпросното изображение може да съдържа котка. От необработени визуални данни на ниво пиксел, машината връща концепция от по-висок ред – „котка“ – въз основа на последователното добавяне и класифициране на тези отделни компоненти. Изображението на Фиг. 2 показва това прогресивно разбиране в контекста на човешкото разпознаване на лицето.



Фиг. 2

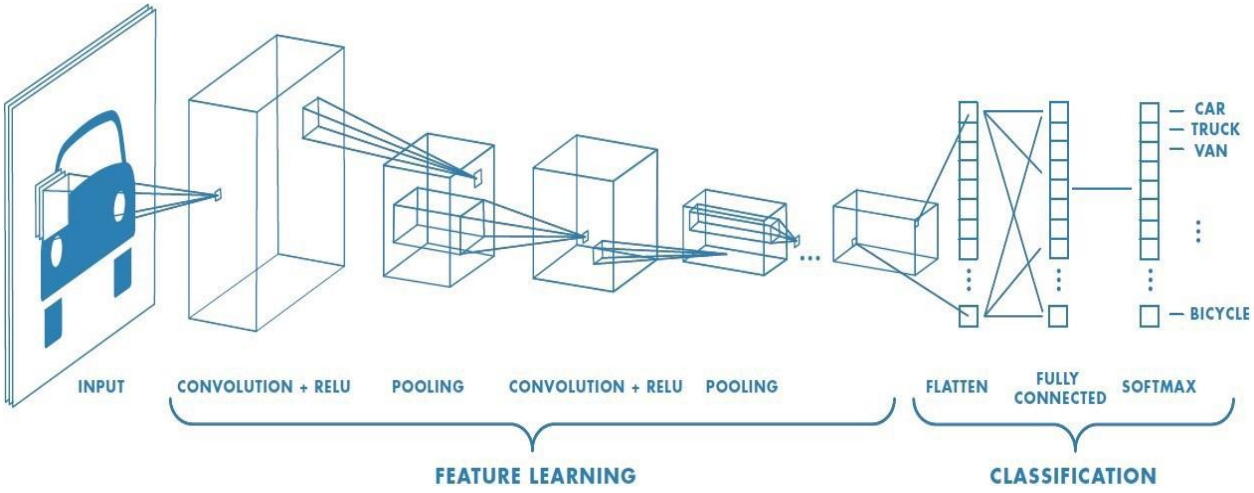
Разликата е, че традиционните системи за зрение включват човека, който казва на машината какво трябва да има, вместо обратният случай, където алгоритъма за дълбоко обучение автоматично извлича характеристиките за това какво има на изображението. Подходът "отдолу нагоре" е значително по-ефективен за определени видове проблеми с анализа на изображенията, много от които използваме често в ежедневието си. При диагностициране на тъканна проба обаче тези разлики стават ценни. Малките, незабележими колебания в плътността на пикселите могат да означават ранното начало на болестта рак - подробности, които дори експерт патолозите е възможно да пропуснат. Човешкото визуално възприятие удря бариера на разделителната способност на изображението при около 2290 пиксела на инч и удря максималност(таван) за възприемане на движението при около 30 кадъра в секунда. Способността на системите за компютърно зрение да работят с точност на ниво пиксел, се повтаря бързо, и изпълнението последователно във времето предлага невероятен потенциал, за увеличаване или надминаване на човешкото възприятие.

CNN или конволюционната невронна мрежа (CNN - convolutional neural network) е клас невронни мрежи за дълбоко обучение. Накратко, можем да си представим CNN като алгоритъм за машинно обучение, който може да приеме входно изображение, да присвои важност (усвоими тегла и отклонения) на различни аспекти/обекти в изображението и да може да ги различава един от друг. Този тип мрежа работи, като извлича характеристики от изображенията. Всяка конволюционна невронна мрежа се състои от следното:

1. Входният слой, който е изображение в сива скала
2. Изходният слой, който е двоичен или мултикласов етикет
3. Скрити слоеве, състоящи се от конволюционни слоеве, слоеве ReLU (ректифицирана линейна единица), обединяващите слоеве и напълно свързана невронна мрежа

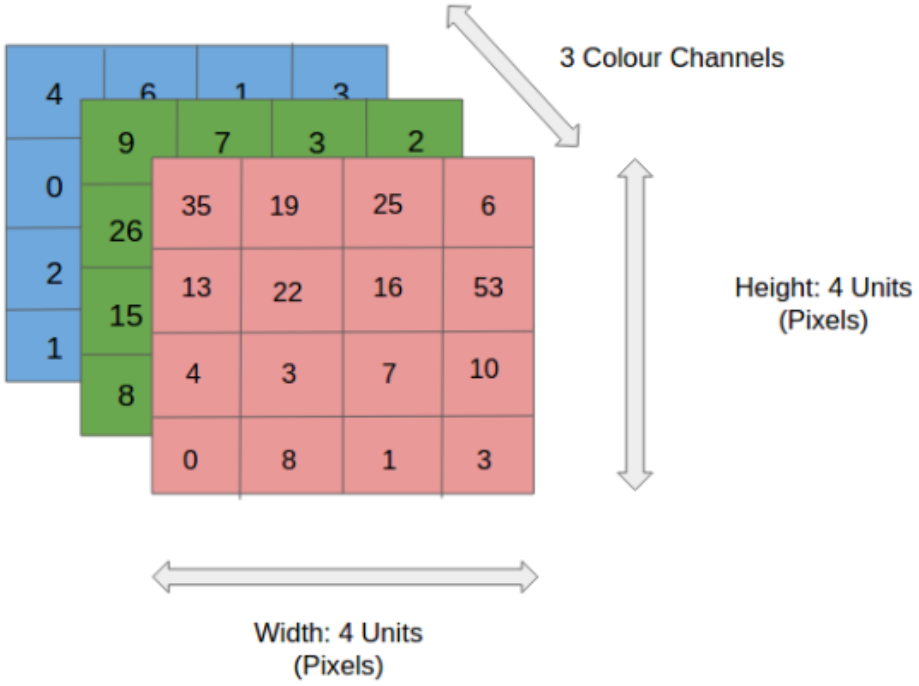
Много е важно да се разбере, изкуствените невронни мрежи, съставени от множество неврони, не могат да извличат характеристики от изображението. Това е мястото, където се появява комбинация от навиващи и обединяващи слоеве. По същия начин, конволюционните и обединяващите слоеве не могат да извършат класификация, следователно се нуждаем от напълно свързана невронна мрежа. Преди да преминем към

концепциите, нека се опитаме да разберем тези отделни сегменти поотделно. Ето един пример от Фиг. 3 който илюстрира процеса на конволюционната невронна мрежа от входни до изходни данни.



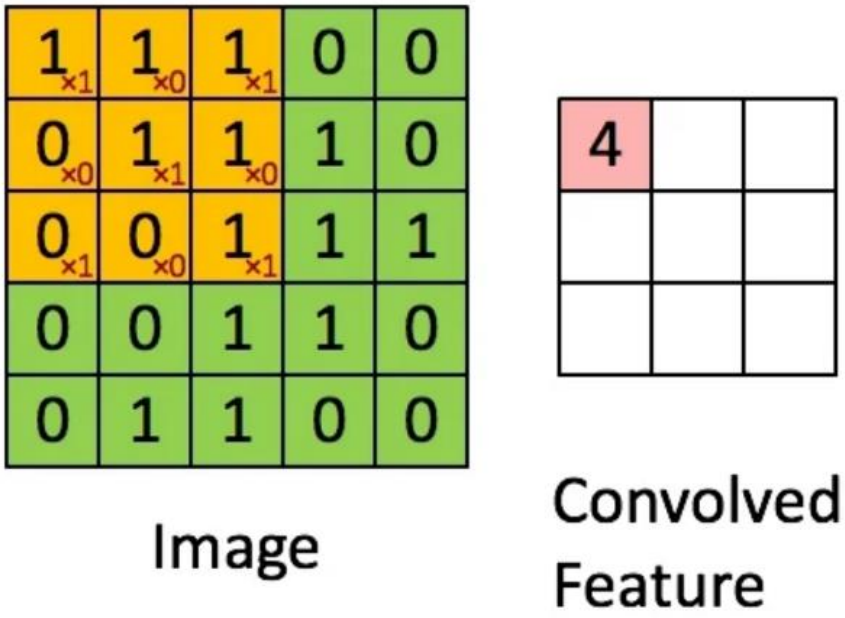
Фиг. 3

Например, ако си представим, че имаме достъп до множество изображения на различни превозни средства, всяко обозначено като камион, кола, микробус, велосипед и т.н. Тогава идеята е да вземем тези предварително етикетирани/класифицирани изображения и да разработим алгоритъм за машинно обучение, който е в състояние за приемане на ново изображение на превозно средство и да го класифицира в правилната му категория или етикет. Преди да започнем да изграждаме невронна мрежа, трябва да разберем, че повечето изображения се преобразуват в сива форма, преди да бъдат обработени. Има значение и какви скали ще използваме – дали ще бъдат сиви скали или RGB/цветни изображения.



Фиг. 4 - RGB цветни канали на изображение

Всяко цветно изображение има три канала, т.е. червено, зелено и синьо, както е показано на Фиг. 4. Има няколко такива цветови пространства като сивата скала, CMYK, HSV, в които може да съществува изображение. Предизвикателството с изображенията с множество цветни канали е, че имаме огромни обеми от данни, с които да работим, което прави процеса изчислително интензивен. В други светове мислете за това като за сложен процес, при който невронната мрежа или всеки алгоритъм за машинно обучение трябва да работи с три различни данни (RGB стойности в този случай), за да извлече характеристики на изображенията и да ги класифицира в съответните им категории. С други думи може да погледнем на това като на сложен процес, при който невронната мрежа или всеки алгоритъм за машинно обучение трябва да работи с три различни данни (RGB стойности в този случай), за да извлече характеристики на изображенията и да ги класифицира в съответните им категории. Ролята на CNN е да намали изображенията във форма, която е по-лесна за обработка, без да губи функции, критични за доброто прогнозиране. Това е важно, когато трябва да направим алгоритъма възможен за мащабиране до масивни набори от данни.



Фиг. 5

Изображението на Фиг. 5 илюстрира стойността на сгънатата характеристика, когато ядрото се прилага към входното изображение. Разбираме, че данните за обучение се състоят от изображения в сивата скала, които ще бъдат вход за слоя на свъртане за извличане на характеристики. Конволюционният слой се състои от едно или повече ядра с различни тежести, които се използват за извличане на характеристики от входното изображение. Да кажем, че в примера по-горе работим с ядро (K) с размер 3 x 3 x 1 (x 1, защото имаме един цветен канал във входното изображение), с тегла, очертани по-долу. Когато плъзгаме ядрото върху входното изображение (да речем, че стойностите във входното изображение са интензитети на сивата скала) въз основа на теглата на ядрото, в крайна сметка изчисляваме характеристики за различни пиксели въз основа на техните околни/съседни пикселни стойности. Например, когато ядрото се прилага върху изображението за първи път, както е илюстрирано на фигурата по-горе, получаваме стойност на характеристика, равна на 4 в сгънатата матрица на характеристиките. Процесът на изместване на ядрото се нарича stride и в нашият пример ядрото ще се измести 9 пъти. Когато използваме операция със стойност на стъпка 1 (Non-Strided), имаме нужда от 9 итерации, за да покрием цялото изображение. Конволюционната невронна мрежа научава тежестта на тези ядра самостоятелно. Резултатът от тази операция е карта на характеристиките, която основно открива характеристики от изображенията, вместо да разглежда всяка една стойност на

пиксела. Резултатът от тази операция е карта на характеристиките, която основно открива характеристики от изображенията, вместо да разглежда всяка една стойност на пиксела.

4. Описание на проблема

В този курсов проект е разгледан класически проблем в машинното обучение и е реализирано решене което се основава на прилагане на техники от дълбокото обучение разгледани в точка 3 от проекта.

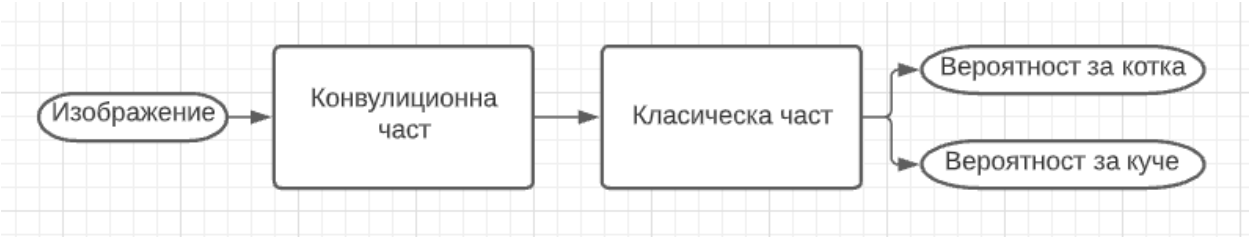
4.1 Дефиниране на разглеждания проблем

Класическият проблем разгледан във този курсов проект е класификационен проблем при който трябва да се синтезира система, която чрез прилагане на техники от дълбокото обучение, да може, с голяма точност, да групира подадената информация на входа й.

В конкретният казус, системата ще бъде обучена да класифицира изображения на кучета и котки. За тази цел са подбрани 3000 фотографски изображения на кучета и котки, заснети в разнообразни ъгли, пози и при различна светлина. Тези изображения ще се използват за обучаване и в последствие, верификация на работоспособността на синтезираната система.

4.2 Модел на решението

Системата която ще се предложи за решение на този проблем ще представлява синтез и в посредством реализация на конволюционна невронна мрежа.



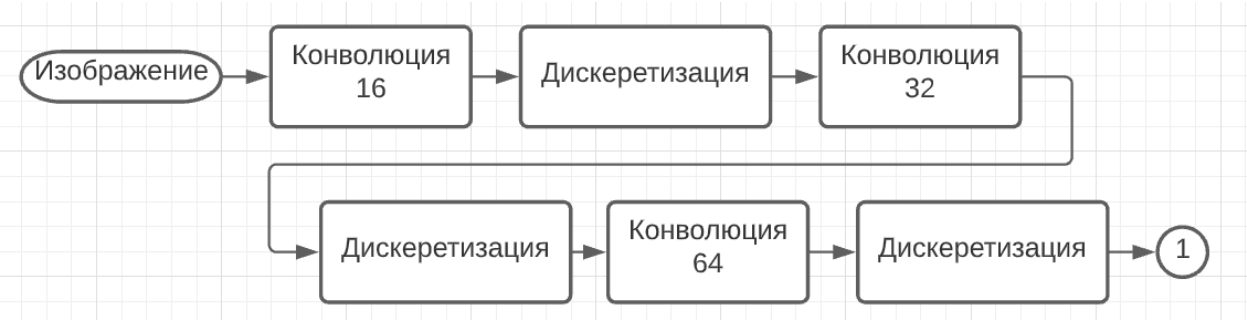
Фиг. 6

Обобщена блок схема на невронната мрежа е представена на Фиг. 6. На входа на мрежата се поддава изображение. На изхода резултата е представен чрез вероятности. Вероятността изображението да принадлежи към класът на кучетата и вероятността изображението да принадлежи към класът на котките Ако се съберат двете вероятности, то резултатът им винаги трябва да 1-ца.

Моделът на конволюционната мрежа показан на Фиг. 6 може условно да се раздели на два блока „Конволиционна част“ и „Класическа част“. Блокът който е наречен

„Конволиционна част“ съдържа слоеве които са присъщи за обработката на изображения във една невронна мрежа. Блокът който е наречен „Класическа част“ представя невронната мрежа в класическият си вид. Тук информацията извлечена от „Конволиционна част“ се обработва и решението се взима.

Схемата на блокът „Конволиционна част“ от Фиг. 6, е представена на Фиг. 7.



Фиг. 7

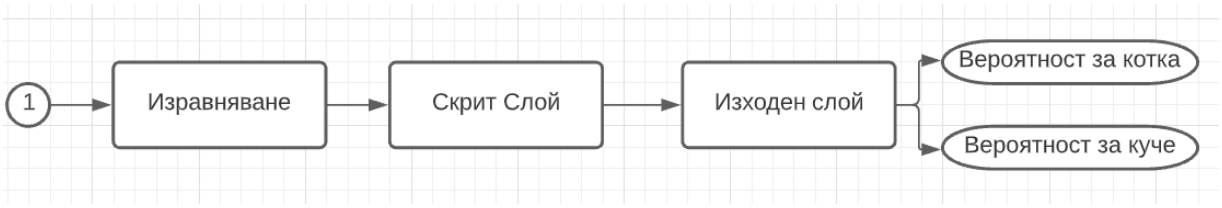
Тук схемата се състои от два вида блока. Блок „Конволюция“ и блок „Дискретизация“. Блоковете „Конволюция“ представляват конволюционните слоеве в невронната мрежа. Чрез тях се извършва процесът конволюция или филтриране на изображението. Числата във всеки един от блоковете показва броят на филтрите които ще се приложат върху данните на входа. Обикновено броят на филтрите за конволюция се определя опитно или чрез използването на задълбочени понятия в дълбокото обучение и компютърното зрение. Тук се използват стандартни стойности, които са установени че дават задоволителни резултати при този и подобни проблеми. Разбира се, възможно е да се изпробват и други стойности и да се експериментира, за да се открият определени конфигурации които биха дали по-добри резултати от разработеното в този проект. Това влиза по-скоро в сферата на оптимизация на невронните мрежи и компютърното зрение и излиза от предмета на този проект.

Другият блок в схемата на Фиг. 7, блок „Дискретизация“, е вид слой от използван в конволюционните невронни мрежи. Неговата функция е да редуцира пикселите във изображението на входа му. Тук се използват методи от класическата обработка на сигнали. По специфично входното изображение се дискредитира и квантува с по-голяма стъпка от дефинираната (down-sampling). Резултантното изображение (или матрица) е с по-малко

пиксели от входното. В конкретният случай на този проект, блок „Дискретизация“ намалява броят на пикселите в изображението, подадено на входа му, два пъти. Това се прави се цел да се редуцира обемът информация информацията във изображението (матрицата) и позволява на следващите слоеве от мрежата да се „фокусират“ основно върху извлечените признаци.

Във всички слоеве, изпълняващи конволюцията, е използвана ReLU (Rectified Linear Activation Function). Това е най-често използваната функция за активация на отделните неврони в една класическа невронна мрежа. В много учебници и статии тази функция се препоръчва да се използва при конволюционни слоеве, тъй като съкращава времето за обучение на мрежата и дава по-добри резултати в дългосрочен план.

Схема на „Класическата част“ на конволюционната мрежа е показана на Фиг. 8.



Фиг. 8

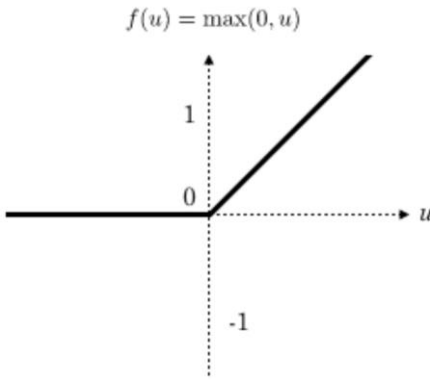
От тук, системата може да се разглежда като класическа невронна мрежа, подобни модели се използват за класификация и групиране на данни. Намират широко приложение във медицината (класификация на туморни образования), икономиката (предвиждане дали даден инструмент на стоковата борса ще се вдигне или ще падне) и във много други среди където информацията не е представена про формата на изображения.

Първият блок, блок „Изравняване“ има за цел да превърне в двумерната матрица на изхода от конволюционната част във едномерен масив или вектор за по-нататъшна обработка. Едномерният масив е стандартният начин по който са представени данните когато се обработват от класическата невронна мрежа.

Блокът „Скрит слой“ или скритият слой (hidden layer) е най-широко използваният слой в класическата невронна мрежа. В този слой се крие „интелигентността“ на невронната

мрежа. Математически този слой (както и невронната мрежа) може да се представи като линейно параметрично уравнение, като броят на параметрите се определя от броят на невронните в скриптят слой за веки вход на слоя. Това е така, понеже един неврон взима за вход всеки елемент от входният вектор, подаден на слоя. След това всеки входен елемент се умножава с тегловен коефициент и резултатите се събират в неврона. Това се прави за всеки неврон от слоят. Резултатите получени от невроните се групират като елементи на вектор и се поддават на следващият слой. Тегловните коефициенти на отделните неврони нямат специфично зададени стойности. Те се определят при обучението на мрежата.

Проблемът в работата на невроните описани в предният параграф е, че така дефинирани, биха могли да описват само линейни функции. Това е валидно при всяка възможна конфигурация на слоя при всички възможни тегловни коефициенти. Проблемът се линейните функции че те са трудно приложими в реалният свят където процесите в почти всички случаи са нелинейни (или близо до линейни). Това прави невронната мрежа не способна да оцени и класифицира реални събития. Поради този проблем, към изходът на всеки неврон се прилага активационна функция. Целта тук е да се използва нелинейна активационна функция с такава крива с която, на теория, невронната мрежа да е в състояние да може да опише абсолютно всякаква възможна нелинейна фукания (чрез наслагване) при всички възможни комбинации от слоеве, брой неврони в тях и тегловни коефициенти. За това в конкретният модел се използва ReLU функцията, чиято крива е дефинирана на Фиг. 9.



Фиг. 9

Последният блок дефиниран като „Изходен слой“ на Фиг. 8. Това практически е скрит слой който има за цел за изведе и форматира резултата от приложените обработки в

невронната мрежа. За конкретната цел се използва Sigmoid activation function. Тази активационна функция позволява на модела да изведе, при класификации, каква е вероятността входният запис да принадлежи на всеки един от дефинираните класове. В конкретният случай това дава вероятностите дали подаденото изображение на входа на системата съдържа котка или куче.

4.3 Подход за обучение на модела

Обучените на невронна мрежа представлява процесът при който, тегловните коефициенти се настройват, спрямо входните записи, предвидени за обучение. Целта тук е тегловните коефициенти да се настроят по такъв начин че, моделът да може успешно да класифицира или оцени нови записи.

При обработката на всеки запис при обучението може да се дефинират 3 етапа:

- Forward pass – при който записът се пуска през невронната мрежа и се изважда резултатът от класификацията или оценката.
- Error correction – тук резултатът изведен от моделът се сравнява с предварително определен резултат и се изчислява коефициентът на грешка.
- Back propagation – след получаване на коефициентът на грешка тегловните коефициенти във модела се променят по такъв начин така че грешката да се минимизира.

Поради това при обучението на невронната мрежа трябва да е избере алгоритъм за оптимизация. Целта на този алгоритъм за оптимизация е да се намери при кои тегловни коефициенти грешката се свежда до минимум.

В този проект избраният метод за оптимизация е Adam оптимизатора. Този метод за оптимизация е широко използван и намира голямо приложение в машинното обучение.

5. Реализация

В тази точка от проекта ще се покажи практическата реализация на подобна система. Разработено е програмно решение, базирано на езикът Python, и използвайки функционалностите на Keras приложният интерфейс (API).

Реализацията ще се раздели на три етапа. При първият етап ще се заредят изображенията и ще се оформи обучаващите множество както и множеството за верификация. След това ще се представи структурата на модела и ще се дефинират отделните слоеве, както и техните параметри. Тук ще се опише и методът, както и параметрите за протичане на обучението. Накрая зададеният метод за обучение ще се изпълни върху моделът. Този процес отнема сравнително дълго време (близо 1 час). Тук ще се представят и графически резултатите от обучението.

5.1 Зареждане на данните

```
trainGen = kr.preprocessing.image.ImageDataGenerator(
                        rescale = 1./255.,
                        rotation_range = 40,
                        width_shift_range = 0.2,
                        height_shift_range = 0.2,
                        shear_range = 0.2,
                        zoom_range = 0.2,
                        horizontal_flip = True)

testGen = kr.preprocessing.image.ImageDataGenerator(rescale = 1.0/255.)
trainData = trainGen.flow_from_directory(TRAIN_DIR, batch_size = 20, class_mode = 'binary', target_size = (150, 150))
testData = testGen.flow_from_directory(TEST_DIR, batch_size = 20, class_mode = 'binary', target_size = (150, 150))
```

Фиг. 10

На Фиг. 10 зареждаме изображенията за трениране и за тестване. Това става с помощта на ImageDataGenerator класа. Този клас създава така наречен „генератор“ обект който позволява да се зареждат изображения. Генераторът може да се конфигурира за да приложи предварителни трансформации върху зареждането изображение. В случая той оразмерява изображението така че стойностите на всички пиксели да са между 0 и 1 (разделя на 1/255). Върху тренирящият генератор се добавят трансформации за завъртане, обръщане, мащабиране и др. с цел усложняване на входните данни.

При генераторът за верификация други операции освен оразмеряване на стойностите на пикселите не се правят.

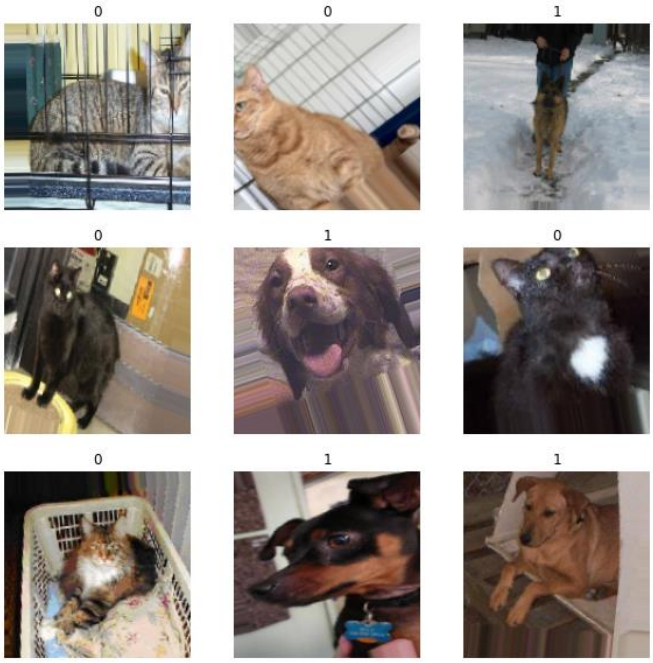
След създаване на генераторите, за всеки генератор се извиква метода flow_from_directory. Този метод приема път към директория като задължителен аргумент. Структурата на тази директория трябва да съдържа директории с изображения, като имената на директориите стават етикети на заредените изображения. В конкретният случай в тази директория ще се съдържат две под-директории, като едната е наименуване „dog" и съдържа всички изображения с кучета, а другата е наименувана „cat" и съдържа всички изображения с котки. Това позволява на всяко изображение на куче да се прикачи етикета „dog" а на всяко изображение с котка етикета „cat". Допълнителни аргументи които метода приема са:

- target_size - размерът на изображенията, в този случай е 150x150.
- class_mode – това се използва за кодиране на етикета, като символните низове (имената от директориите) се превръщат в поредни стойности. Примерно за етикета „cat" може да е 1 а за етикета „dog" може да е 0.
- batch_size – този параметър практически групира изображенията във групи по 20.

Този метод се прилага за както за обучаващата така и за информацията за верификация. След прилагането му информацията за обучение се запазва в променливата trainData, а информацията с която ще се верифицира модела в testData. На Фиг. 11 и Фиг. 12 може да се види видът на заредената информация за обучение.

```
plt.figure(figsize=(10,10))
for (images, labels) in [trainData[0]]:
  for i in range(9):
    ax = plt.subplot(3, 3, i+1)
    plt.imshow((images[i]*255).astype('uint8'))
    plt.title(int(labels[i]))
    plt.axis('off')
```

Фиг. 11



Фиг. 12

5.2 Описание на модела

Описанието на моделът следва блок схемата на Фиг. 6, Фиг. 7 и Фиг. 8 от точка 4.2. Реализирането му става като се използва Sequential моделът на Keras, показан на Фиг. 13. Sequential моделът е най-простият начин за дефиниране на модел в Keras, като при него слоевете са свързани последователно. Той взима само един задължителен аргумент и това е списък с слоевете на модела.

```
model = kr.models.Sequential([
    kr.layers.Conv2D(16, (3,3), activation='relu', input_shape=(150, 150, 3)),
    kr.layers.MaxPooling2D(2,2),
    kr.layers.Conv2D(32, (3,3), activation='relu'),
    kr.layers.MaxPooling2D(2,2),
    kr.layers.Conv2D(64, (3,3), activation='relu'),
    kr.layers.MaxPooling2D(2,2),
    kr.layers.Flatten(),
    kr.layers.Dense(512, activation='relu'),
    kr.layers.Dense(1, activation='sigmoid')
])

model.summary()
```

Фиг. 13

Класовете дефиниращи слоевете се съдържат в layers модула и значението им е:

- layers.Conv2D – това практически е блок „Конволюция“ от Фиг. 7. Като първият задължителен аргумент е броят на филтрите които ще се приложат. Вторият аргумент обозначава размерът на филтрите, в случая 3x3. activation, специфицира функцията за активиране на невроните в слоя, тук е използвана ReLU функцията още показана на Фиг. 9. input_shape, задава размерът на едно изображение. Това се прави само за входният слой, тъй като следващите се структурират спрямо входния.
- kr.layers.MaxPooling2D – това е блок „Дискретизация“ от Фиг. 7. Като стъпката на дискретизация е 2 по хоризонтал и 2 по вертикал.
- kr.layers.Flatten – това е блок „Изравняване“ от Фиг. 8.
- kr.layers.Dense – са съответно блок „Скрит слой“ и блок „Изходен слой“ от Фиг. 8. Като тук се използва само един скрит слой с 512 неврона и ReLU функция за активация. За изходният слой, тук е предпочетено да се използва само един неврон, понеже етикетите са взаимно изключващи се и се дефинира следната зависимост. Ако примерно невронът на изхода си изведе стойността, 0.7 то това се приема че има 70% процента вероятност да е котка (която е 0 в този случай) и 30% вероятност да е куче (1 в случая). Ако на изхода има 0.2 то тогава вероятностите са 20% - котка, 80% - куче.

Тук също се извиква и метода summary. Който извежда информация за модела. Това е показано на Фиг. 14. Тук може би най-нетеренната информация е че в моделът се съдържат почти 9.5 милиона параметъра.

Model: "sequential"		
Layer (type)	Output Shape	Output Shape
conv2d (Conv2D)	(None, 148, 148, 16)	448
max_pooling2d (MaxPooling2D)	(None, 74, 74, 16)	0
conv2d_1 (Conv2D)	(None, 72, 72, 32)	4640
max_pooling2d_1 (MaxPoolling2D)	(None, 36, 36, 32)	0
conv2d_2 (Conv2D)	(None, 34, 34, 64)	18496
max_pooling2d_2 (MaxPoolling2D)	(None, 17, 17, 64)	0
flatten (Flatten)	(None, 18496)	0
dense (Dense)	(None, 512)	9470464
dense_1 (Dense)	(None, 1)	513
Total params:	9 494 561	
Trainable params:	9 494 561	
Non-trainable params:	0	

Фиг. 14

За обучението на този модел ще се използва Адам оптимизатора представен в точка 4.3. Той се задава начина представен на Фиг. 15.

```
model.compile(optimizer=kr.optimizers.Adam(learning_rate=0.001),
              loss='binary_crossentropy',
              metrics = ['accuracy'])
```

Фиг. 15

5.3 Обучение и представяне на резултатите

Обучението при Keras може да стане заедно с верификацията като на всеки цикъл (epoch) се изпълнява обучение и верификация. От верификацията след това да настроят параметрите за следващият цикъл. Метода за обучение е показан на Фиг. 16.

```
history = model.fit(
    trainData,
    validation_data=testData,
    steps_per_epoch=100,
    epochs=50,
    validation_steps=50)
```

Фиг. 16

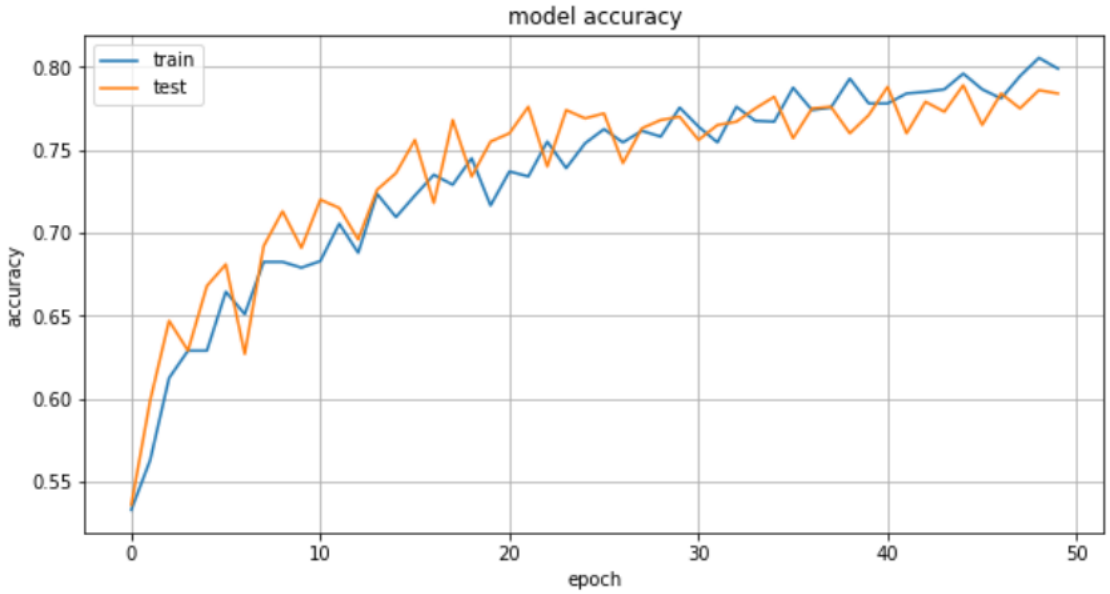
Тук се използва метода fit. Той приема като задължителен аргумент данните за обучение. След това се подават и данните за верификация. Следващите аргументи биха изисквали повече обяснение.

Всеки цикъл (epoch) може да се представи като редица от под-цикли. Във всеки под цикъл се взима една група (batch) изображения като всяко изображение преминава паралелно през модела, изчислява се общата грешка и се правят необходимите корекции по-параметрите. Броят на под-циклите в един цикъл се дава с аргумента steps_per_epoch. В случая той е 100. Това има връзка със броят на изображенията в една група. В точка 5.1 това беше зададено на 20. Тъй като общо всички изображения за обучение са 2000, то тогава групите за обучение ще са 100. Идеята тук е в един цикъл да се изредят всички изображения за обучение. След всеки цикъл за обучение се прави цикъл за верификация, който също се състои от под-цикли, обаче в този случай не се прави корекция на параметрите при всяка група а само се измерва точността на модела. Броят на верификационните под-цикли се дава с аргумента validation_steps и е 50. След верификацията параметрите на модела може да се

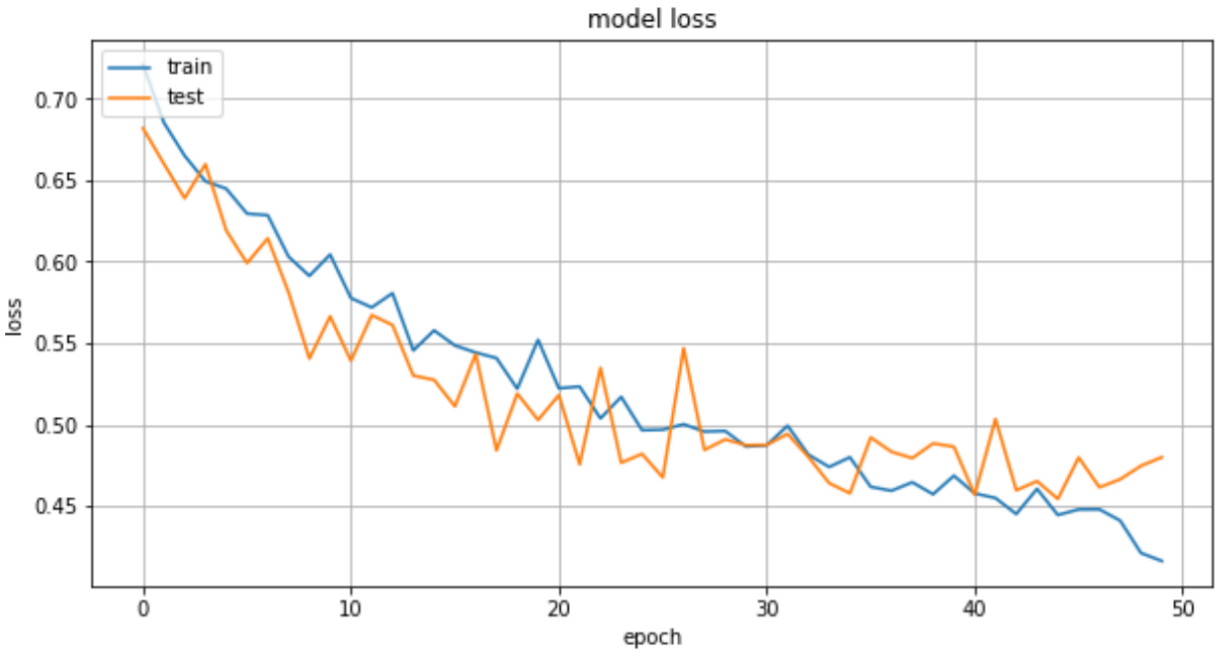
до настроят, както и да се промени начина на обучение при следящите цикли. Пълният брой на цикли за обучение на модела се подава с аргумента epochs и в този случай е 50.

След обучението, от последният цикъл за верификация се извежда следната оценка на точността: 79.9%.

Графични резултати за точността на модела след всеки цикъл може да се видят на Фиг. 17. Като тук е дадена оценка и след обучението в цикъла и след верификацията на цикъла. Така може да се види какво би трябвало да е очакваната точност след протичане на цикъл и каква е действителната оценка след верификация на цикъла. Оценката след верификация в повечето случаи е по-ниска от първоначалната. И във двете криви може да се забележи увеличение на точността със всеки изминал цикъл, като подобрението става все по минимално със всеки изминал цикъл. Това означава че дори и се увеличи броят на циклите, резултатът няма да се промени значително (за сметка на това времето за обучение ще е много голямо). Решението тук би било да се потърси или опитно да се установи каква промяна в структурата на модела или алгоритъма за обучение би била най-благоприятна за крайната оценка. Друго нещо което може да се промени с цел увеличаване на точността би било да се увеличи обемът на информацията за обучение. Тук може да се използва трик при който изображения от обучаващото множество се дублират но се добавят и допълнителни трансформации като ротация, мащабиране и др.



Фиг. 17



Фиг. 18

На Фиг. 18 са представени загубите на модела при всеки цикъл. Тук загуби се дефинират като брой грешно класифицирани изображения, към пълният брой изображения в цикъла. Той дава подобна информация като оценката на точността от Фиг. 17. Разликата между точността и загубите е че загубите се използват като входни аргументи към функцията която оптимизира модела, докато точността е крайна оценка и е предназначена за да се покаже точността на модела пред в по-представителен вид.

6. Примери за приложение

Засичането и разпознаването на изображения намира голямо приложение в съвременният свят. Примери за това може да са разпознаване на човешки лица през охранителни камери в реално време с цел да се намери издирван престъпник. Откриване на заболявания по чертите на лицето на човека.

FaceNet е алгоритъм разработен от Google. С това те успяват да направят нов рекорд за точност достигайки до 99.63% (± 0.0009) използвайки популярният „Labeled Faces in the Wild“ сет от изображения. Тази функционалност е вградена във приложението Google Photos и му дава възможност да подрежда снимките на потребителя както и да отбелязва личностите които са на дадена снимка както и да позволява търсене чрез името на даден човек (в кои снимки той може да бъде видян.

Конволюционните невронни мрежи могат да се използват за разработката на автопилот за автомобили. Тук се използват и допълнителни техники за откриване на обекти в реално време но за самото разпознаване, класификация и идентификация на обекта се използва модел подобен на разгледания (и в пъти по оптимизиран).

Компютърното зрение намира широко приложение и в сферата на здравеопазването. Тук то може да се използва от за откриване и разпознаване на най-малките детайли от изображения на рентген, ядрено магнитен резонанс, ехограф и други изследвания. Такива детайли биха били трудно уловени от човешкото око.

7. Използвана литература

[1] Empower, „Каква е разликата между изкуствен интелект и машинно обучение,“ [Онлайн]. Available: https://www.digital.bg/kakva-e-razlikata-mezhdu-izkustven-intelekt-i-mashinno-obuchenie-article673640.html.

[2] „Какво представлява машинното обучение?,“ [Онлайн]. Available: https://bg.itpedia.nl/2018/04/05/wat-is-machine-learning/.

[3] N. Babich, „What Is Computer Vision & How Does it Work? An Introduction,“ [Онлайн]. Available: https://xd.adobe.com/ideas/principles/emerging-technology/what-is-computer-vision-how-does-it-work/.

[4] M. Taylor, „Computer Vision with Convolutional Neural Networks,“ [Онлайн]. Available: https://medium.com/swlh/computer-vision-with-convolutional-neural-networks-22f06360cac9.

[5] A. Das, „Convolution Neural Network for Image Processing — Using Keras,“ [Онлайн]. Available: https://towardsdatascience.com/convolution-neural-network-for-image-processing-using-keras-dc3429056306.

[6] M. Sanguineti, „Cats VS Dogs Convolutional Classifier,“ [Онлайн]. Available: https://towardsdatascience.com/cats-vs-dogs-convolutional-classifier-44ec04c8eb7a.

[7] E. Allibhai, „Building a Convolutional Neural Network (CNN) in Keras,“ [Онлайн]. Available: https://towardsdatascience.com/building-a-convolutional-neural-network-cnn-in-keras-329fbbadc5f5.

[8] A. Thevenot, „Conv2d: Finally Understand What Happens in the Forward Pass,“ [Онлайн]. Available: https://towardsdatascience.com/conv2d-to-finally-understand-what-happens-in-the-forward-pass-1bbaafb0b148.

[9] J. Brownlee, „A Gentle Introduction to the Rectified Linear Unit (ReLU),“ [Онлайн]. Available: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/.

[10] Y. VERMA, „A Complete Understanding of Dense Layers in Neural Networks,“ [Онлайн]. Available: https://analyticsindiamag.com/a-complete-understanding-of-dense-layers-in-neural-networks/.

[11] „How does Sigmoid activation work in multi-class classification problems,“ [Онлайн]. Available: https://datascience.stackexchange.com/questions/39264/how-does-sigmoid-activation-work-in-multi-class-classification-problems.

Приложение

```
!rm -rf *
!curl -Lk https://storage.googleapis.com/mledu-
datasets/cats_and_dogs_filtered.zip --output /tmp/download.zip
!unzip -q /tmp/download.zip -d .
!mv cats_and_dogs_filtered dataset



import os
import os.path as path
import tensorflow as tf
import tensorflow.keras as kr
import matplotlib.pyplot as plt



BASE_DIR = 'dataset'
TRAIN_DIR = path.join(BASE_DIR, 'train')
TEST_DIR = path.join(BASE_DIR, 'validation')



model = kr.models.Sequential([
    kr.layers.Conv2D(16, (3,3), activation='relu', input_shape=(150, 150,
3)),
    kr.layers.MaxPooling2D(2,2),
    kr.layers.Conv2D(32, (3,3), activation='relu'),
    kr.layers.MaxPooling2D(2,2),
    kr.layers.Conv2D(64, (3,3), activation='relu'),
    kr.layers.MaxPooling2D(2,2),
    kr.layers.Flatten(),
    kr.layers.Dense(512, activation='relu'),
    kr.layers.Dense(1, activation='sigmoid')
])

model.summary()

model.compile(optimizer=kr.optimizers.Adam(learning_rate=0.001),
              loss='binary_crossentropy',
              metrics = ['accuracy'])



trainGen = kr.preprocessing.image.ImageDataGenerator(
                                    rescale = 1./255.,
                                    rotation_range = 40,
                                    width_shift_range = 0.2,
                                    height_shift_range = 0.2,
                                    shear_range = 0.2,
                                    zoom_range = 0.2,
                                    horizontal_flip = True)
testGen = kr.preprocessing.image.ImageDataGenerator(rescale = 1.0/255.)
trainData = trainGen.flow_from_directory(TRAIN_DIR, batch_size = 20,
class_mode = 'binary', target_size = (150, 150))
testData = testGen.flow_from_directory(TEST_DIR, batch_size = 20, class_mode
= 'binary', target_size = (150, 150))
```

```
x = trainData[0]
print(len(x))
print(type(x))
print(len(trainData)*x[0].shape[0])

# batches = [(images, labels)]
# images = ndarray 20

plt.figure(figsize=(10,10))
for (images, labels) in [x]:
  for i in range(9):
    ax = plt.subplot(3, 3, i+1)
    plt.imshow((images[i]*255).astype('uint8'))
    plt.title(int(labels[i]))
    plt.axis('off')


history = model.fit(
    trainData,
    validation_data=testData,
    steps_per_epoch=100,
    epochs=50,
    validation_steps=50)


plt.figure(figsize=(10,5))
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.grid()
plt.show()


plt.figure(figsize=(10,5))
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.grid()
plt.show()
```