



Софийски университет „Св. Климент Охридски“

Факултет по математика и информатика

Катедра „Компютърна информатика“

ДИПЛОМНА РАБОТА

на тема

**„Система за разпознаване на пол и възраст на човек от
видео в реално време“**

Дипломант: **Димитър Тодоров Чаушев**
Магистърска програма: **Изкуствен интелект**
Факултетен номер: **25336**

Научен ръководител:
проф. д-р Мария Нишева

Съдържание

1.	Въведение.....	3
1.1	Компютърно зрение	3
1.2	Преглед на задачата	3
1.3	Цели на дипломната работа.....	3
2.	Конволюционни невронни мрежи.....	4
2.1	Архитектура на визуалния кортекс	5
2.2	Конволюционни слоеве	6
	Филтри	8
	Наслагване на картите на свойства (feature maps).....	9
	Изисквания за памет	10
2.3	Свързващи слоеве (Pooling layers)	11
2.4	Архитектури	11
	LeNet-5	12
	AlexNet (2012)	13
	VGG-16 (2014).....	14
	Inception	15
	ResNet.....	15
	Xception	16
3.	Откриване на лица в изображение.....	17
3.1	Haar Cascade Face Detector	17
3.2	Подход, базиран на невронна мрежа	19
3.3	Други подходи	20
4.	Избор и подготовка на набора от тренировъчни данни	21
4.1	IMDB-WIKI.....	21
4.2	Предварителна обработка на IMDB-WIKI	22
4.3	UKTFace	24
4.4	Допълнителна обработка след обединяване на данните	26
5.	Предсказване на възраст от изображение на лице	28
5.1	Избор на модел	29
5.2	Обучение.....	30
6.	Предсказване на пол от изображение на лице	32

6.1	mini-Xception	32
6.2	Традиционна конволюционна невронна мрежа.....	34
7.	Използван софтуер и работа на приложението	36
7.1	Използван софтуер и хардуер	36
7.2	Файлова структура и стартиране на приложението	36
8.	Проведени експерименти	39
9.	Бъдещо развитие	42
10.	Заключение	44
	Литература	45

1. Въведение

1.1 Компютърно зрение

Компютърното зрение традиционно е една от най-активните изследователски области за прилагане на дълбоко обучение, защото да виждаме – за нас хората, а и за много животни е задача, която изпълняваме без усилие, но за компютрите е предизвикателство.

Компютърното зрение е много широка област, обхващаща различни начини за обработване на изображения и невероятно разнообразие от приложения, които варират от имитиране на човешкото зрение, до създаване на изцяло нови категории визуални способности. Пример за последното е приложение, което разпознава звукови вълни от вибрациите, които те предизвикват върху обекти, видими на видео [1]. Повечето научни изследвания в областта на компютърното зрение не се фокусират върху толкова екзотични приложения (които дори е трудно да си представим), а по-скоро са концентрирани върху малък набор от задачи, опитващи се да репликират човешките възможности като разпознаване на обекти или оптично разпознаване на знаци.

1.2 Преглед на задачата

Една от задачите в областта на компютърното зрение е разпознаването на пола и възрастта на човек по снимка на неговото лице. Автоматичното разпознаване на тези две характеристики намира множество приложения, особено след възхода на социалните мрежи. Подобна функционалност е много подходяща за интеграция със системи за наблюдение и охрана. Един такъв пример е системата за масов надзор в Китай, където има инсталирани над двеста милиона камери. Системата в реално време успява да разпознае пола, възрастта, дори и облеклата на хората [2].

Друго възможно приложение е в сферата на рекламите – представете си електронно табло с камера, което спрямо възрастовата група и пола на хората, минаващи покрай него, показва различни реклами.

1.3 Цели на дипломната работа

Целта на дипломната работа е да се създаде система, която разпознава пола и възрастта на човек по неговото лице от видео в реално време.

Тъй като всеки кадър от видеото представлява единично изображение, задачата се свежда до три подзадачи:

- откриване на лица в изображение;
- разпознаване на пол от изображение на лице;
- разпознаване на възраст от изображение на лице.

И за трите задачи ще се използват конволюционни невронни мрежи, но знаейки, че системата трябва да работи в реално време, моделите трябва да са достатъчно олекотени, като в същото време да дават задоволителни резултати.

След направено проучване върху други подобни разработки се стигна до заключение, че резултатите, които могат да бъдат приети като задоволителни, са:

- поне 0.8 MAE грешка за регресионен модел за предсказване на възраст [3];
- поне 93% точност за класификационен модел за предсказване на пол [4];
- по-малко от 0.05 секунди време за едно предсказване, за да няма осезаемо забавяне и системата да може да работи в реално време.

В следващата секция ще разгледаме теорията зад конволюционните невронни мрежи и защо точно те са най-подходящи за решаването на поставените задачи.

2. Конволюционни невронни мрежи

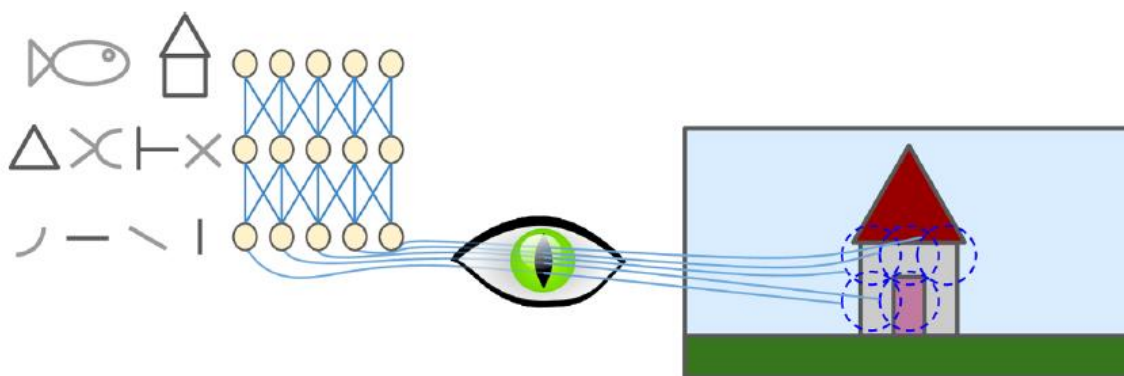
Едва от скоро компютрите са в състояние надеждно да изпълняват на пръв поглед тривиални задачи като това да открият кученце в картина или да разбират говор. Но защо тези задачи са толкова лесни за хората? Отговорът се крие във факта, че възприятието до голяма степен се осъществява извън сферата на нашето съзнание, в рамките на специализирани зрителни, слухови и други сензорни модули в мозъка ни. Докато сетивната информация достигне нашето съзнание, тя вече е украсена с по-абстрактни характеристики. Например, когато погледнете снимка на сладко кученце не можете да изберете да **не** видите кучето, нито да **не** забележите колко е сладко. Нито можете да обясните как разпознавате кучето. За вас е просто очевидно. Възприятието изобщо не е тривиално и за да го разберем, трябва да си изясним как работят сетивните сензори.

Конволюционните невронни мрежи се появяват при изследване на зрителния кортекс на мозъка и се използват при разпознаването на изображения от 80-те. През последните години, благодарение на увеличението на изчислителната мощ и количеството налични данни, конволюционните невронни мрежи успяват да постигнат свръхчовешки постижения върху някои сложни визуални задачи. Те захранват услуги за търсене на изображения, самоуправляващи се автомобили, системи за автоматична класификация на изображения и други. Още повече, конволюционните невронни мрежи не са ограничени до визуалното възприятие. Те намират успешно приложение и при много други задачи, като гласово разпознаване и обработка на естествен език. Ще се съсредоточим обаче върху визуалните приложения, тъй като това е темата на дипломната работа.

2.1 Архитектура на визуалния кортекс

David H. Hubel и Torsten Wiesel правят серии от експерименти върху котки през 1958 г. [5] и 1959 г. [6], като стигат до повратни прозрения относно структурата на визуалния кортекс (през 1981 г. авторите получават Нобелова награда за работата си). В частност те показват, че много от невроните във визуалния кортекс имат малко локално поле на възприятие, което означава, че те реагират само на визуални стимули, намиращи се в ограничен регион от визуалното поле (*Фигура 1*). Полетата на възприятие на различните неврони могат да се припокриват и заедно да изградят цялото зрително поле.

Авторите показват, че някои неврони реагират само на изображения на хоризонтални линии, други реагират на такива с вертикални и т.н. Те са забелязали и, че някои неврони имат по-големи полета на възприятие и реагират на по-сложни модели – комбинация от по-прости такива. Тези наблюдения водят до идеята, че невроните от по-високо ниво са базирани на изхода на съседните си неврони от по-ниско ниво. Такава мощна архитектура е в състояние да открие всякакви сложни модели в каквато и да е зона от визуалното поле.

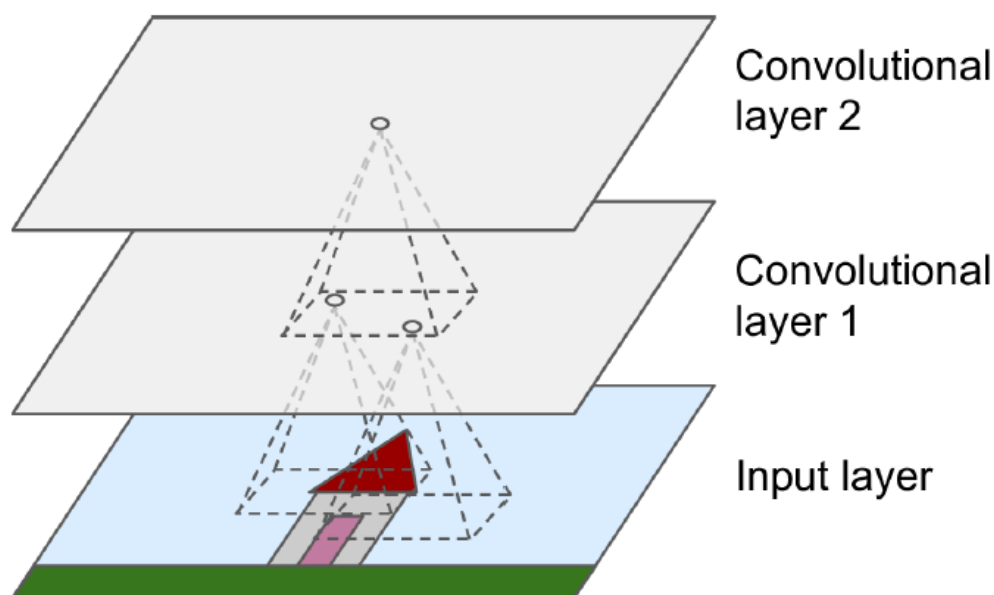


Фигура 1. Биологичните неврони във визуалния кортекс реагират на специфични модели в малки региони от визуалното поле. Докато визуалните сигнали си проправят път през последователните мозъчни модули, невроните реагират на по-сложни модели в по-голямо поле. [7]

Тези проучвания на визуалния кортекс са вдъхновението на неокогнитрона [8], представен през 1980 г., който в последствие еволюира до това, което сега наричаме конволюционни невронни мрежи. Важно събитие е и дисертацията на Yann LeCun от 1998 г. [9], в която представя известната *LeNet-5* архитектура – широко използвана от банки за разпознаване на ръкописно написани чекове. Тази архитектура използва вече познатите по онова време напълно свързани слоеве и активиращата функция сигмоид, но също така представя два нови градивни елемента – конволюционните слоеве и обединяващите слоеве (pooling layers).

2.2 Конволюционни слоеве

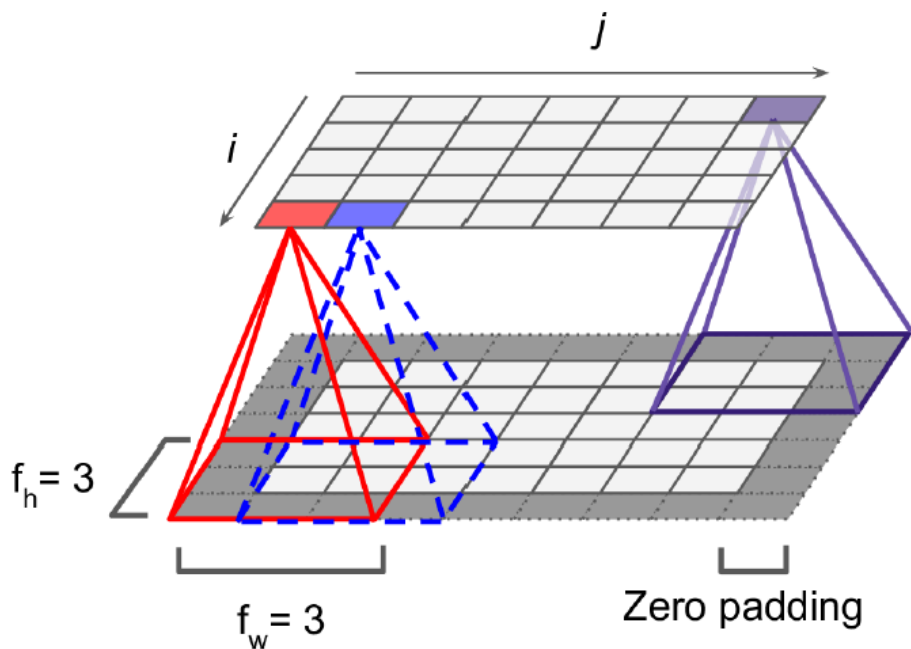
Най-важният градивен елемент на конволюционните невронни мрежи е конволюционният слой. Невроните в първия конволюционен слой не са свързани с всеки един пиксел от входното изображение, а само с пикселите от техните полета на възприятие (*Фигура 2*). На свой ред, всеки неврон от втория конволюционен слой е свързан само с неврон, намиращ се в малък правоъгълник от първия конволюционен слой. Тази архитектура позволява на мрежата да се концентрира върху малки елементи от по-ниско ниво в първия скрит слой, като в следващите слоеве ги сглобява в по-големи елементи от по-високо ниво и т.н.



Фигура 2. Конволюционни слоеве с правоъгълни полета на възприятие. [7]

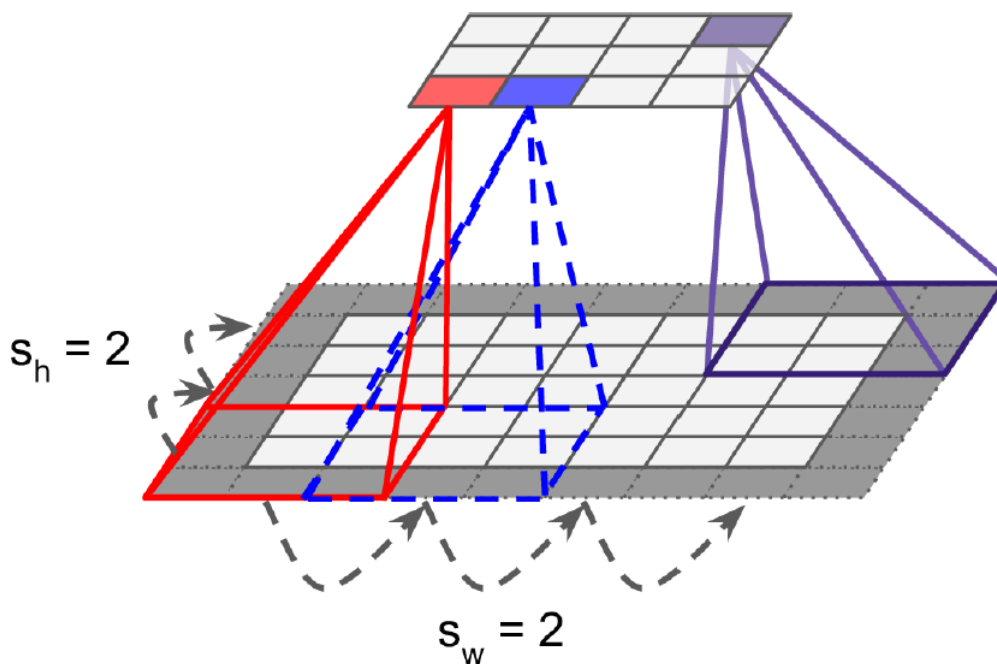
При конволюционните невронни мрежи всеки слой е представен в двумерно пространство и невроните от всеки слой се характеризират с координати за ред и колона.

Неврон, намиращ се на ред i , колона j от даден слой е свързан с изхода на неврони от предишен слой, намиращи се на редове от i до $i + f_h - 1$, и колони j до $j + f_w - 1$, където f_h и f_w са височината и ширината на полето на възприятие. За да имат слоевете една и съща височина и ширина като предходните им, често срещана практика е да се добавят нули около входните пиксели, както се вижда на *Фигура 3*. Това се нарича нулево уплътняване (zero padding).



Фигура 3. Връзка между слоеве и нулево уплътняване. [7]

Възможно е да се свърже голям входен слой с много по-малък слой като полетата на възприятие се изместват на разстояние едно от друго. Това значително намалява изчислителната сложност на модела. Прехвърлянето от едно поле към следващото се нарича стъпка/крачка (stride). На *Фигура 4*, слой с размер 5×7 (плюс нулево уплътняване) е свързан със слой с размер 3×4 , използвайки поле на възприятие 3×3 и стъпка 2. Невронът, намиращ се на ред i , колона j от горния слой е свързан с изходите от невроните от предишния слой, намиращи се на редове $i \times s_h$ до $i \times s_h + f_h - 1$ и колони $j \times s_w$ до $j \times s_w + f_w - 1$, където s_h и s_w са вертикалната и хоризонталната стъпки.

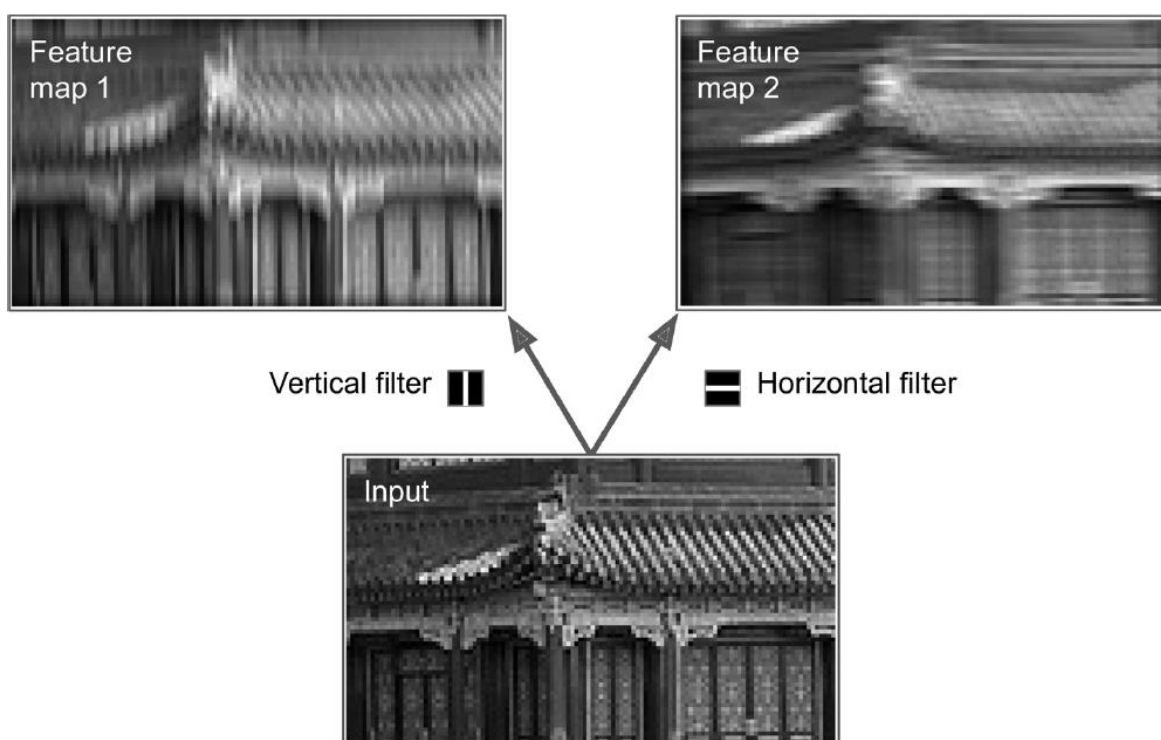


Фигура 4. Намаляване на размерността като се използва стъпка с размер 2. [7]

Филтри

Теглата на невроните могат да бъдат представени като малки изображения с размера на полето на възприятие. На *Фигура 5* са показани два възможни набора от тегла, наречени филтри (или конволюционни ядра). Първият е представен като черен квадрат с вертикална бяла черта по средата. Невроните, използващи тези тегла, ще игнорират всичко от тяхното поле на възприятие с изключение на централната вертикална линия. Вторият филтър е черен квадрат с хоризонтална бяла линия по средата. Отново, невроните, използващи тези тегла, ще игнорират всичко освен централната хоризонтална линия.

Ако всички неврони в даден слой използват филтъра с вертикални линии и на мрежата се подаде долното изображение от *Фигура 5*, изходът ще бъде лявото изображение – вертикалните бели линии са засилени, докато всичко останало е замъглено. По подобен начин дясното изображение се получава ако всички неврони използват филтъра с хоризонталната линия. Изходът от слой, чиито неврони използват един и същ филтър, се нарича карта на свойства (feature map), която акцентира на зоните от изображението, които активират филтъра най-много. Разбира се, не е нужно тези филтри да се дефинират ръчно. Вместо това, по време на обучение конволюционният слой сам ще научи най-полезните филтри за своята задача и слоевете над него ще се научат да ги комбинират в по-сложни модели.

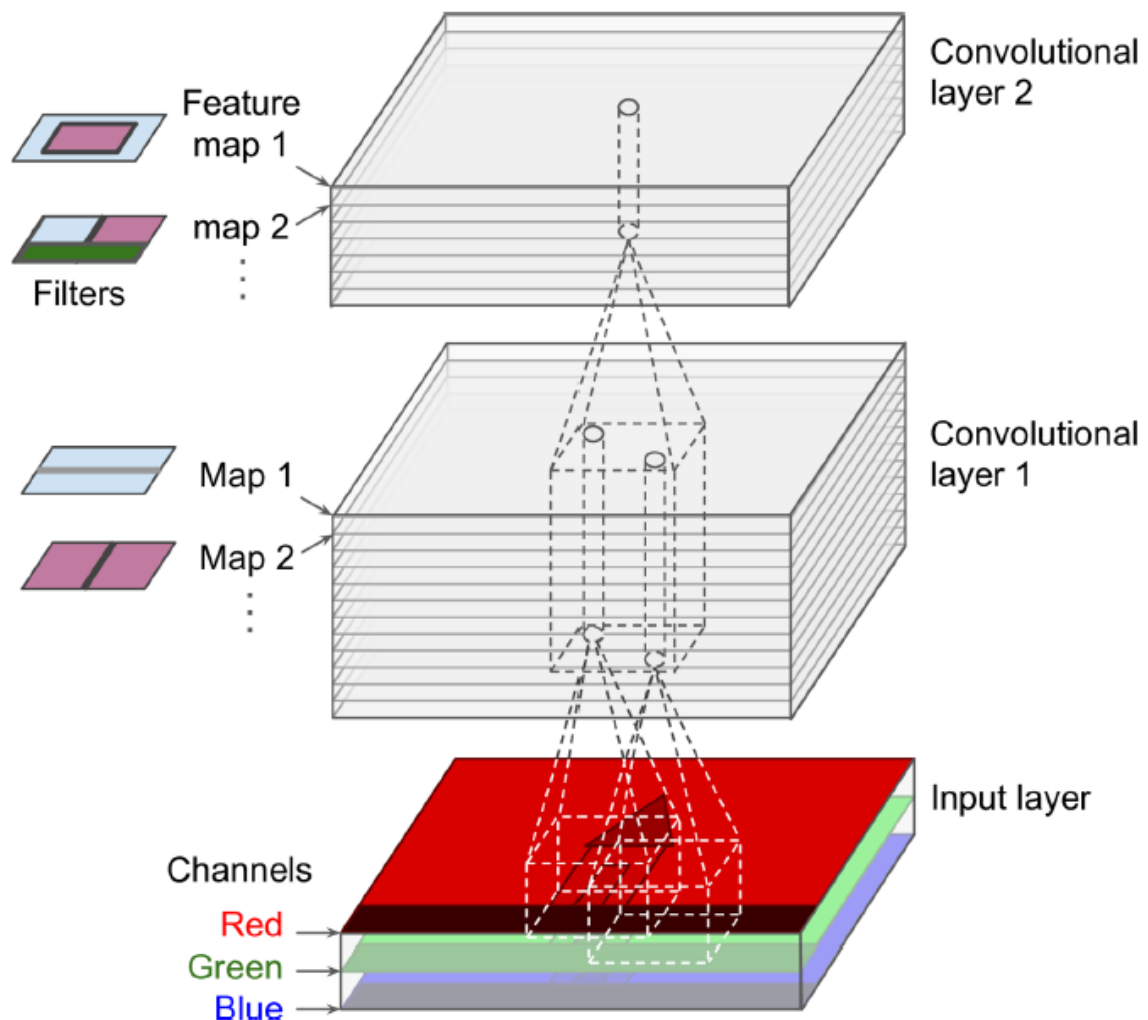


Фигура 5. Прилагане на два различни филтъра за да получим карти на две свойства. [7]

Наслагване на картите на свойства (feature maps)

В действителност конволюционният слой има множество филтри (Фигура 6) и произвежда по една карта със свойства за всеки филтър. Има по един неврон за всеки пиксел във всяка карта и всички неврони от дадена карта споделят едни и същи параметри (тегла и отклонение). Това значително намалява броя на параметрите в един конволюционен модел. След като веднъж конволюционната мрежа се научи да разпознава даден шаблон на определена локация, то тя ще може да го разпознава и на всяка друга локация. В сравнение, щом една обикновена дълбока невронна мрежа се научи да разпознава даден шаблон на определена локация, то тя ще може да го разпознава само и единствено на тази локация.

Входните изображения имат допълнителни под-слоеве – по един за всеки цветен канал. По принцип има три канала – червен, зелен и син. Черно-белите изображения имат само един канал.



Фигура 6. Конволюционни слоеве с множество карти на свойства и изображение с три цветни канала.
[7]

За пресмятането на изхода на даден неврон от конволюционен слой се използва следното уравнение:

$$z_{i,j,k} = b_k + \sum_{u=0}^{f_h-1} \sum_{v=0}^{f_w-1} \sum_{k'=0}^{f_{n'}-1} x_{i',j',k'} \cdot w_{u,v,k',k}$$

където $i' = (i \times s_h + u)$ и $j' = (j \times s_w + v)$

В това уравнение:

- $z_{i,j,k}$ е изходът на неврон, намиращ се на ред i и колона j в карта на свойствата k от конволюционния слой;
- s_h и s_w са вертикалната и хоризонталната стъпки;
- $x_{i',j',k'}$ е изходът от неврон от предишния слой, намиращ се на ред i' , колона j' , карта на свойствата k' (или канал k' ако предишния слой е бил входният);
- b_k е отклонението (bias term) на карта k от текущия слой;
- $w_{u,v,k',k}$ е свързващото тегло между някой неврон от карта k на текущия слой и вход, намиращ се на ред u , колона v и карта k' .

Изисквания за памет

Проблем при конволюционните невронни мрежи е, че конволюционните слоеве изискват огромно количество оперативна памет. Това е особено вярно по време на трениране, защото връщането при обратното разпространение на грешката (backpropagation) има нужда да знае всички междинни стойности, които са сметнати по време на преминаването напред (forward pass).

Нека вземем за пример конволюционен слой с филтри с размер 5×5 , които произвеждат 200 карти на свойства с размер 150×100 , стъпка 1 и нулево уплътняване. Ако входът е цветно изображение (3 канала) с размер 150×100 пиксела, тогава броят на параметрите е $(5 \times 5 \times 3 + 1) \times 200 = 15\,200$ (+1 е за отклонението/bias). Всяка от 200-те карти има 150×100 неврона и всеки от тези неврони трябва да сметне притеглена сума на своите $5 \times 5 \times 3 = 75$ входа, което прави 225 000 000 умножения на числа с плаваща запетая. Не е толкова зле, колкото при напълно свързания слой (fully connected layer), но все пак е доста за изчисляване. Освен това, ако картите на свойства (feature maps) са представени, използвайки 32 битови числа с плаваща запетая, тогава изходът на конволюционния слой ще окупира $200 \times 150 \times 100 \times 32 = 96\,000\,000$ бита, което е 12 MB RAM. И това е само за едно изображение. Ако обработваме изображенията в партии от по 100, тогава този слой ще използва 1.2 GB RAM.

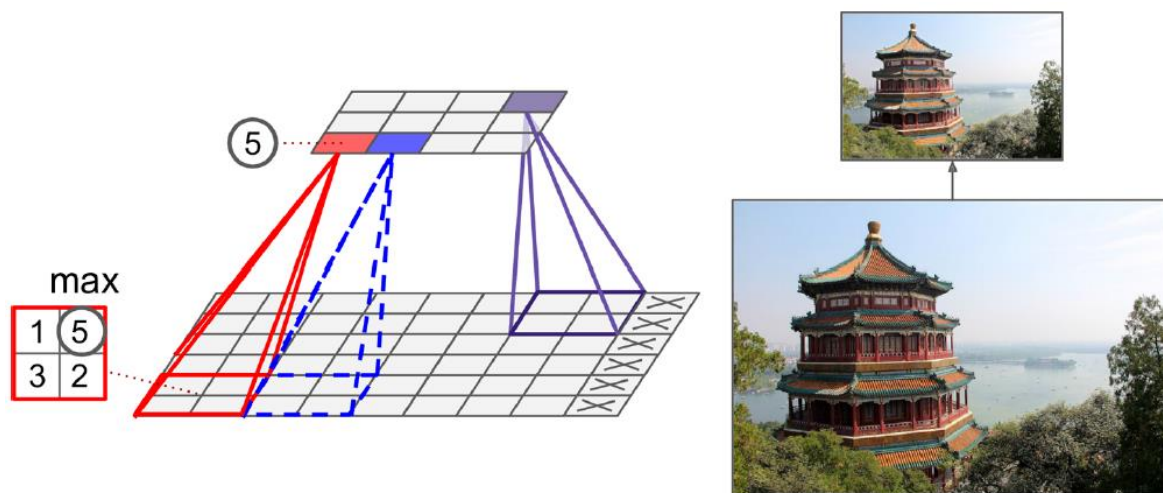
При предсказване оперативната памет за един слой може да бъде освободена веднага, след като следващият слой бъде изчислен, така че ни трябва толкова памет, колкото за два слоя едновременно. Но по време на трениране всички изчисления от

преминаването напред са нужни и при връщането назад. И общата необходима памет е паметта, нужна за всички слоеве.

2.3 Свързващи слоеве (Pooling layers)

Целта на свързващите слоеве е да свие (намали размера на) входното изображение с цел редуциране на изчислителното натоварване, използваната памет и броя на параметрите.

Също както при конволюционните слоеве, всеки неврон от свързващия слой е свързан с изходите на ограничен брой неврони от предходния слой. Размера, стъпката и уплътняването са ръчно дефинирани. Свързващият слой няма тегла. Единственото, което прави е да агрегира входовете, като използва агрегираща функция като *max* или *mean*. *Фигура 7* показва свързващ слой, използващ функцията *max*, което е най-често срещаният тип свързващ слой. На примера от тази фигура се използва свързващо ядро (pooling kernel) с размери 2×2 , стъпка 2 и няма нулево уплътняване. Само максималната стойност от полето на възприятие (receptive field) ще бъде взета за следващия слой, а останалите стойности ще бъдат изпуснати. В случая, стойностите са 1, 5, 3 и 2, така, че само стойността 5 ще стигне до следващия слой. Заради стъпката от 2, изходното изображение ще е с двойно по-малък размер от входното (закръглено надолу, тъй като не се използва уплътняване).

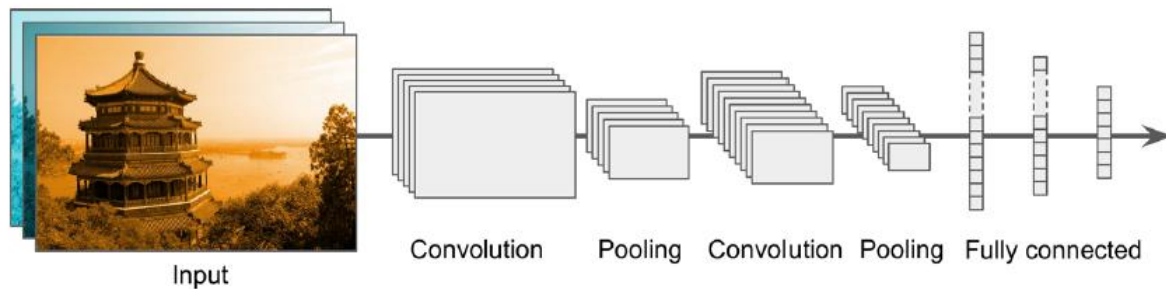


Фигура 7. Максимален свързващ слой (2×2 , стъпка 2, без уплътняване). [7]

2.4 Архитектури

Типичната архитектура на конволюционна невронна мрежа, показана на *Фигура 8*, е подреждане на няколко конволюционни слоя (всеки следван от активационен

слой – най-често ReLU, *Фигура 9*), след това свързващ слой, след това още няколко конволюционни слоя (+ReLU) следвани от друг свързващ слой и т.н.



Фигура 8. Типична архитектура на конволюционна невронна мрежа. [7]

Изображението става по-малко и по-малко докато преминава през мрежата, но също така става по-дълбоко и по-дълбоко (с повече свойствени карти), благодарение на конволюционните слоеве. После се добавя обикновена поточна (feedforward) невронна мрежа, състояща се от няколко напълно свързани слоя (+ReLU) и финален слой, който прави предсказанието (например, softmax слой, който извежда вероятността за принадлежност към всеки клас, *Фигура 9*).

Name	Formula	Derivative	Graph	Range
ReLU (rectified linear unit)	$\text{relu}(a) = \max(0, a)$	$\frac{\partial \text{relu}(a)}{\partial a} = \begin{cases} 0, & \text{if } a \leq 0 \\ 1, & \text{if } a > 0 \end{cases}$		$(0, \infty)$
softmax	$\sigma_i(\mathbf{a}) = \frac{e^{a_i}}{\sum_j e^{a_j}}$	$\frac{\partial \sigma_i(\mathbf{a})}{\partial a_j} = \sigma_i(\mathbf{a}) (\delta_{ij} - \sigma_j(\mathbf{a}))$ Where δ_{ij} is 1 if $i=j$, 0 otherwise		$(0, 1)$

Фигура 9. Активационни функции ReLU и softmax. ReLU е активационна функция, която взема максимума от 0 и подадената стойност, т.е. връща стойност между 0 и безкрайност. Softmax е активационна функция, която връща вероятността разглеждания обект да принадлежи към всеки от класовете.

През годините са се разработили много вариации на тази фундаментална архитектура, довели до невероятния напредък на областта. Следва да разгледаме най-популярните сред тях.

LeNet-5

Както споменахме по-рано, LeNet-5 [9] е разработена от Yann LeCun през 1998 г. Тя е една от най-простите архитектури (*Фигура 10*). Има 2 конволюционни и 3

напълно свързани слоя или общо 5, откъдето и идва името ѝ. Често срещано е имената на архитектурите да се извеждат от броя на слоевете им. Свързващият слой тогава се е наричал *sub-sampling* слой и е имал тегла, което не е практика при съвременните мрежи. Тази архитектура има около 60 000 параметъра.

Layer	Type	Maps	Size	Kernel size	Stride	Activation
Out	Fully connected	–	10	–	–	RBF
F6	Fully connected	–	84	–	–	tanh
C5	Convolution	120	1×1	5×5	1	tanh
S4	Avg pooling	16	5×5	2×2	2	tanh
C3	Convolution	16	10×10	5×5	1	tanh
S2	Avg pooling	6	14×14	2×2	2	tanh
C1	Convolution	6	28×28	5×5	1	tanh
In	Input	1	32×32	–	–	–

Фигура 10. Архитектура на LeNet-5.

Тя се е превърнала в стандартния шаблон: редуване на конволюционни и свързващи слоеве, завършвайки с един или повече напълно свързани слоя.

AlexNet (2012)

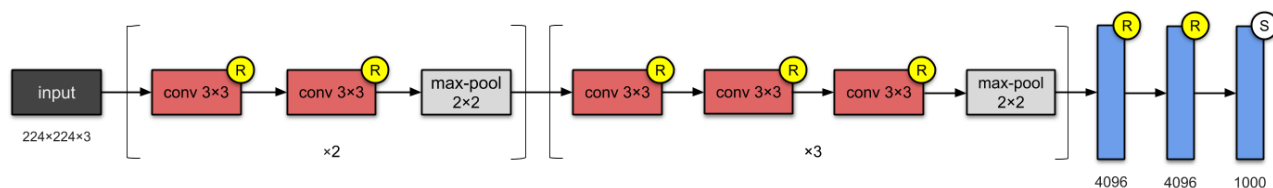
С 60 милиона параметъра, AlexNet [10] има 8 слоя – 5 конволюционни и 3 напълно свързани (Фигура 11). AlexNet просто трупа още слоеве отгоре LeNet-5. По време на представянето на публикацията, авторите посочват, че тяхната архитектура е една от най-големите конволюционни невронни мрежи за това време. Те са първите, използвали Rectified Linear Units (ReLU) като активационна функция.

Layer	Type	Maps	Size	Kernel size	Stride	Padding	Activation
Out	Fully connected	–	1,000	–	–	–	Softmax
F9	Fully connected	–	4,096	–	–	–	ReLU
F8	Fully connected	–	4,096	–	–	–	ReLU
C7	Convolution	256	13×13	3×3	1	same	ReLU
C6	Convolution	384	13×13	3×3	1	same	ReLU
C5	Convolution	384	13×13	3×3	1	same	ReLU
S4	Max pooling	256	13×13	3×3	2	valid	–
C3	Convolution	256	27×27	5×5	1	same	ReLU
S2	Max pooling	96	27×27	3×3	2	valid	–
C1	Convolution	96	55×55	11×11	4	valid	ReLU
In	Input	3 (RGB)	227×227	–	–	–	–

Фигура 11. Архитектура на AlexNet.

VGG-16 (2014)

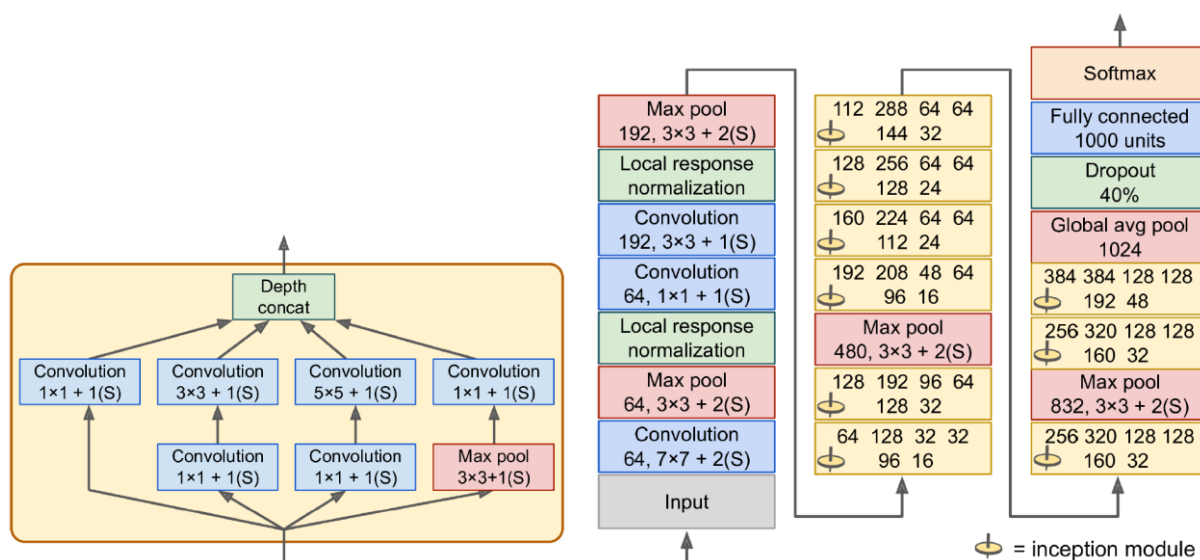
Забелязва се, че конволюционните невронни мрежи стават все по-дълбоки и по-дълбоки. Това е, защото най-лесният начин за подобряването им е увеличаване на размера им. Хората от Visual Geometry Group (VGG) изобретяват VGG-16 [11], която има 13 конволюционни слоя и 3 напълно свързани, използвайки ReLU от AlexNet. Архитектурата на тази мрежа може да се види на *Фигура 12*. Тя наслагва още слоеве над AlexNet и използва по-малки филтри – 2×2 и 3×3 . Състои се от 138 милиона параметъра и заема около 500MB място за съхранение. Има разработен и по-дълбок вариант – VGG-19.



Фигура 12. Архитектура на VGG-16.

Inception

Разработена от Christian Szegedy от Google Research [12]. Представя така наречените Inception модули, които позволяват на архитектурата (Фигура 13) да използва по-ефикасно параметрите си. Има 10 пъти по-малко параметри в сравнение с AlexNet, като в същото време е по-дълбока.

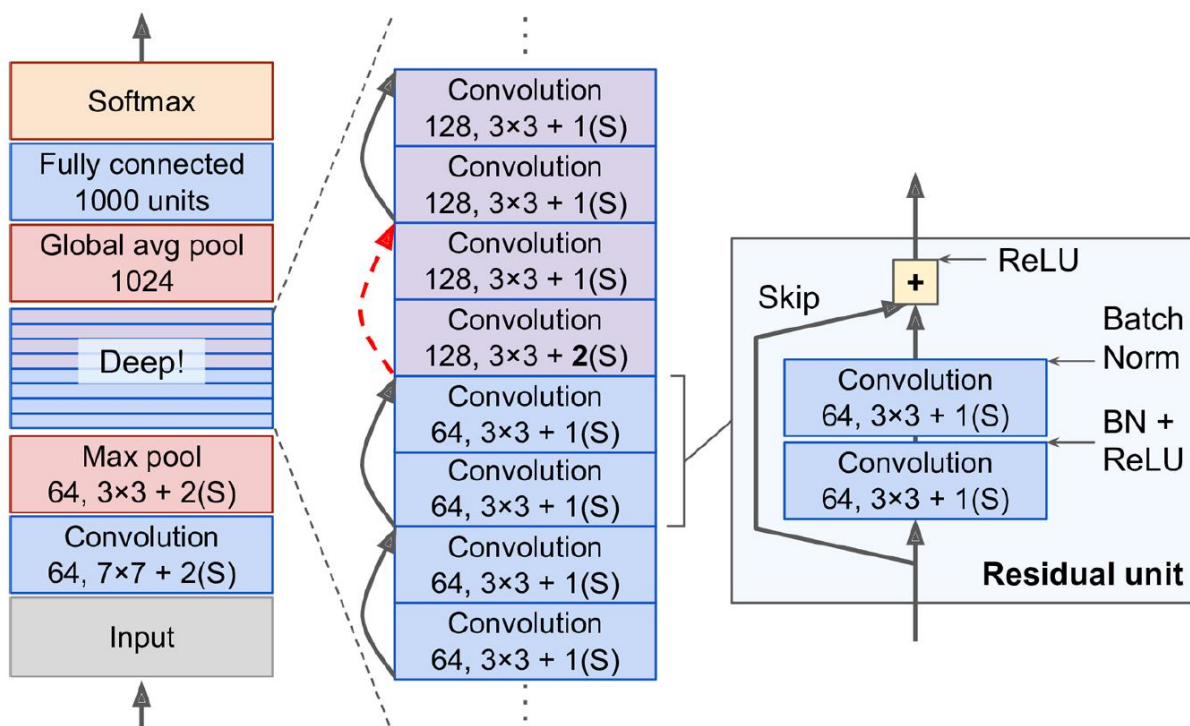


Фигура 13. Вляво – Inception Module. Вдясно – Inception архитектурата.

ResNet

Има варианти с 34, 50, 101 и 152 слоя [13]. Затвърждава модата, че моделите стават по-дълбоки и по-дълбоки, но същевременно с по-малко и по-малко параметри (Фигура 14). За да успеят да тренират толкова дълбока мрежа, използват т.н. преки връзки (*skip connections* или *shortcut connections*). По принцип, по време на трениране на една невронна мрежа, целта е да се научи функция $h(x)$. Ако обаче добавим x към изходния слой (добавяне на *skip connection*), тогава мрежата ще е принудена да моделира функция $f(x) = h(x) - x$. Това се нарича остатъчно учене (*residual learning*). При инициализацията на невронна мрежа, теглата ѝ са близки до 0, така че изходът на мрежата е близък до 0. При добавянето на пряка връзка обаче, мрежата ще произвежда копие на входните си данни. С други думи, това е моделиране на функцията на идентичност. Ако целевата функция е близка до функцията на идентичност (което е доста често), това ускорява процеса на трениране значително.

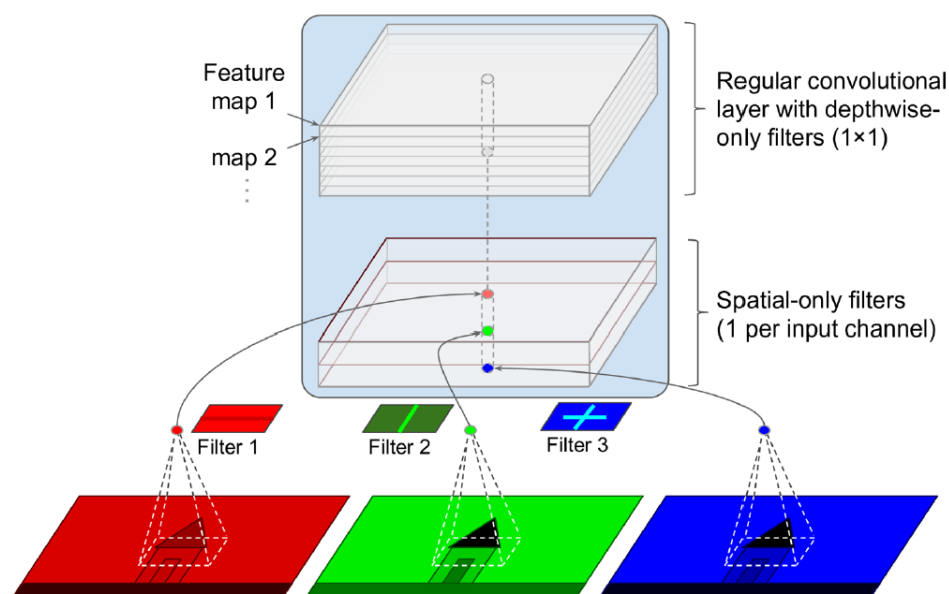
Ако се добавят много преки връзки, мрежата започва да прави прогрес дори преди слоевете да са започнали да се учат.



Фигура 14. ResNet архитектурата.

Xception

Представена е през 2016 г. от François Chollet [14], авторът на библиотеката Keras [15]. Обединява идеите на Inception и ResNet, но замества inception модулите със специален тип слой, наречен *depthwise separable convolution layer* (Фигура 15). Докато обикновените конволюционни слоеве използват филтри, които едновременно улавят пространствени модели (като овал) и обобщаващи модели (като уста + нос + очи = лице), *separable convolution* слойт предполага, че пространствените модели и обобщаващите модели могат да бъдат моделирани отделно. Следователно е съставен от две части – първата прилага пространствен филтър за всяка входна карта на свойствата, след това втората част търси специално за обобщаващи модели – нещо като обикновен конволюционен слой с филтри 1×1 .



Фигура 15. Depthwise separable convolution слой.

Разгледахме архитектурите, довели до огромния успех в областта на компютърното зрение. За съжаление, както ще видим по-късно, поради изчислителни ограничения, повечето от тези невронни мрежи нямаше как да бъдат обучени и обучените мрежии са базирани на LeNet архитектурата. Единствено мрежата за откриване на лица в изображение е базирана на ResNet архитектурата, но тя се използва предварително обучена.

3. Откриване на лица в изображение





Първата от подзадачите, поставени като цел в точка 1.3, е откриване на лица в изображение. Разгледани са четири подхода за решаване на тази задача, като на практика са изпробвани два от тях.

3.1 Haar Cascade Face Detector

Подходът за откриване на лица в изображения, базиран на алгоритъма Haar Cascade е бил “state-of-the-art” подходът в продължение на години след представянето си през 2001 г. от Paul Viola и Michael J. Jones [16].

Haar филтрите са подобни на тези от конволюционните слоеве. Разликата е, че при конволюционните невронни мрежи филтрите на конволюционните ядра се откриват автоматично по време на обучение, а при Haar са ръчно въведени.

Няколко Хаг филтъра са представени на *Фигура 16*. Първите два се използват за откриване на ръбове/краища. Третият открива вертикални линии, а четвъртият най-вероятно се използва за откриване на наклонени линии. Целта на всеки филтър е да открие дадена част от лицето (*Фигура 17*).

-1	-1	5
-1	-1	5
-1	-1	5

5	5	5
-1	-1	-1
-1	-1	-1

-1	5	-1
-1	5	-1
-1	5	-1

5	-1	-1
-1	5	-1
-1	-1	5

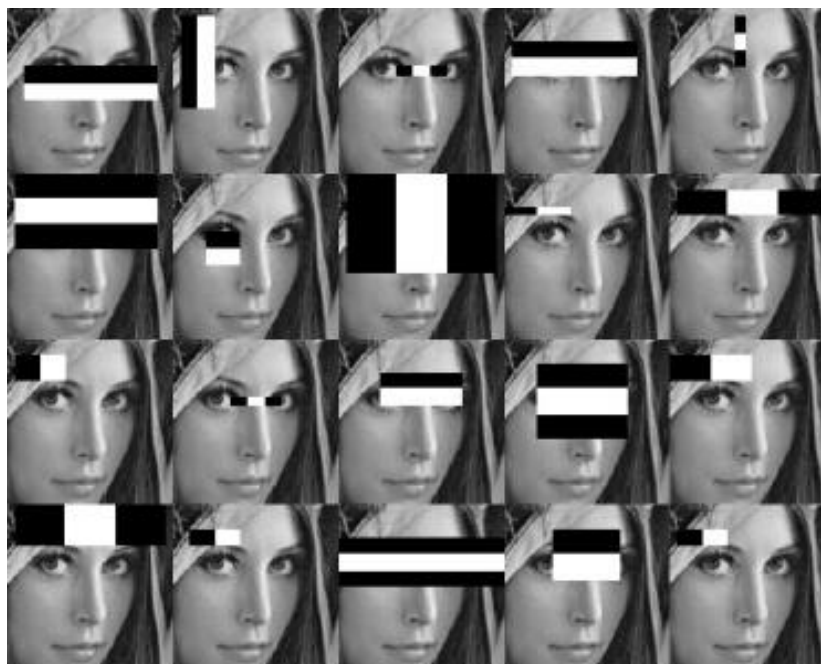
(1)

(2)

(3)

(4)

Фигура 16. Примери за Хаг филтри



Фигура 17. Прилагане на Хаг филтри върху изображение на лице.

За да открива съответните лицеви характеристики, алгоритъмът „плъзга“ всеки филтър по изображението.

Предимствата на подхода са:

- работи почти в реално време върху процесор;
- има проста архитектура;
- намира лица с различни големина.

Недостатъците обаче са:

- прави много грешни предсказания;
- не намира лица, които не са фронтални.

Резултати от експериментите с този подход са показани на *Фигура 18*.



Фигура 18. Тестване на Haar Cascade Face Detector.

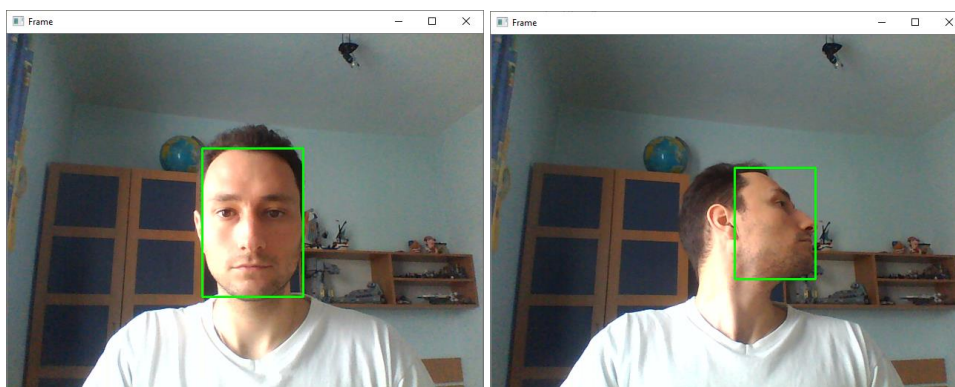
3.2 Подход, базиран на невронна мрежа

Следващият разгледан подход е базиран на **Single-Shot-Multibox detector** [17] и използва по-плътна версия на ResNet архитектурата – ResNet-10. Моделът е обучаван върху изображения от интернет. OpenCV библиотеката [18] предоставя модела и теглата към него.

Това е и моделът, избран за ползване в приложението:

- най-точният от всички проучени подходи;
- работи в реално време върху процесор;
- работи с различни ориентации на лицето (*Фигура 19*);
- открива лица с различна големина.

Превъзхожда Haar Cascade във всяко отношение и то без недостатъци.



Фигура 19. Тестване на модела за откриване на лица, базиран на ResNet-10.

3.3 Други подходи

Другите два проучени подхода са HoG (Histogram of Oriented Gradients) Face detector [19] и Face detector от библиотеката Dlib [20], базиран на конволюционна невронна мрежа [21].

HoG подхода е базиран на машини с поддържащи вектори. Използва 5 филтъра – фронтален, гледащ наляво, гледащ надясно, фронтален, но леко завъртян наляво и фронтален, но леко завъртян надясно. Обучаван е върху 2825 изображения, взети от Labeled Faces in the Wild [22] базата от данни и ръчно маркирани от автора – Davis King.

Предимства:

- най-бързият подход;
- работи добре на фронтални и леко обърнати лица;
- олекотен модел, в сравнение с другите – моделът е вграден директно в header файла [23].

Основните недостатъци са:

- не намира малки лица. Изисква минимум от 80×80 пиксела;
- ограждащата кутия често изрязва част от лицето и част от брадата;
- не работи добре при лица, които са обърнати настрани или гледат нагоре или надолу.

Последния подход е базиран на конволюционна невронна мрежа и използва **Maximum-Margin Object Detector** [24].

Предимства:

- работи при различни ориентации на лицето;
- работи много бързо върху видео карта;
- лесен за трениране.

Недостатъци:

- много бавен върху процесор;
- също като предходния, изисква лице с минимум 80×80 пиксела;
- ограждащата кутийка е дори по-малка, отколкото при HoG метода.

4. Избор и подготовка на набора от тренировъчни данни

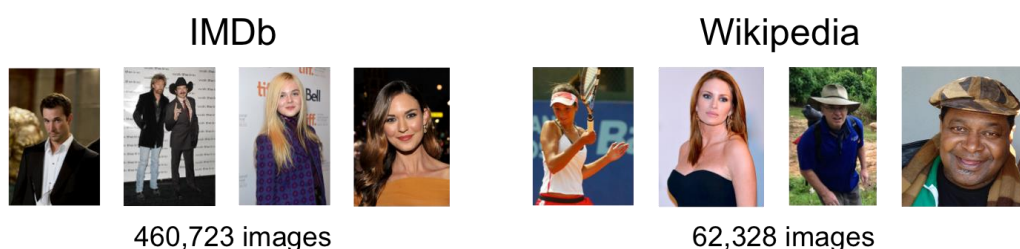
4.1 IMDB-WIKI

За трениране на моделите за разпознаване на пол и възраст е нужна база от данни със снимки на лица, маркирани с пола и възрастта на лицето от снимката.

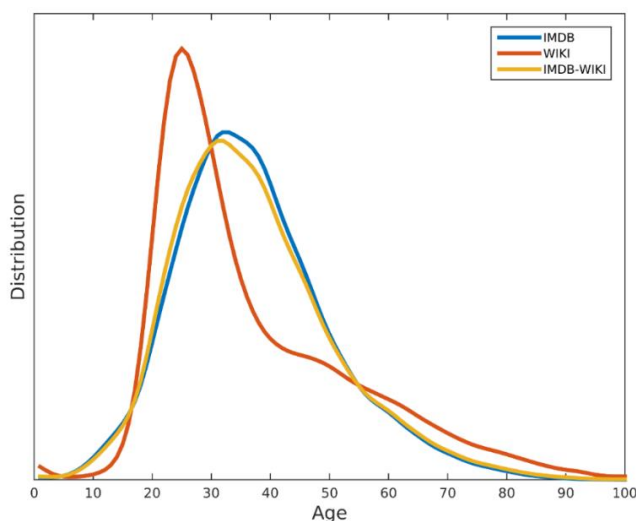
Най-голямата публично достъпна такава база е IMDB-WIKI [25]. Тъй като до тогава публично достъпните такива бази от данни са били много малки, авторите са решили да си създадат собствена и да я направят публично достъпна.

За тази цел са взели имената на най-популярните 100 000 актьори от IMDB [26] и автоматично са свалили датите им на раждане, името, пола и всички снимки с дата на заснемане, свързани със съответния човек. В допълнение са свалили и същия набор от данни за личности от Wikipedia [27]. Предполили са, че снимките, на които има само един човек се отнасят за правилния индивид и са ги маркирали със свалената информация. Успели са да подготвят 460 723 снимки от IMDB и 62 328 снимки от Wikipedia, което прави общо 523 051 снимки.

Примери на снимките от тази база от данни могат да се видят на *Фигура 20*, а как са разпределени снимките спрямо възрастта е показано на *Фигура 21*.



Фигура 20. Примерни снимки от IMDB-WIKI базата.



Фигура 21. Разпределение на данните в IMDB-WIKI базата спрямо възрастта.

За всяка снимка в базата от данни има следната информация:

- **dob**: data of birth или датата на раждане;
- **photo_taken**: годината на заснемане на снимката;
- **full_path**: пътят до файла на снимката;
- **gender**: пол, 0 за женски и 1 за мъжки. *NaN* ако е неизвестно;
- **name**: името на човека;
- **face_location**: локацията на лицето на снимката;
- **face_score**: увереността, с която алгоритъмът е заключил, че има лице на снимката. С *Inf* се маркира, че няма лице на снимката;
- **second_face_score**: увереността, с която алгоритъмът е заключил, че има второ лице на снимката. *NaN* ако няма второ лице;
- **celeb_name** и **celeb_id**: отнасят се само за IMDB базата и носят името и ID-то на човека според IMDB.

За да се изчисли възрастта на човек, е нужно да се извади датата на раждане от датата на заснемане на снимката.

4.2 Предварителна обработка на IMDB-WIKI

Тъй като цялата база е създадена автоматично, голяма част от информацията в нея е грешна или непълна. Предварителната обработка на данните включва:

- премахване на снимки без пол, т.е. **gender** == *NaN*;
- премахване на снимки без лице, т.е. **face_score** == *Inf*;
- премахване на снимки с повече от едно лице, т.е. **second_face_score** != *NaN*;
- премахване на снимки, при които увереността за наличието на лице е ниска, т.е. **face_score** < 3;
- след пресмятане на възрастта, премахване на снимки, при които възрастта е по-малка от 1 или над 100.

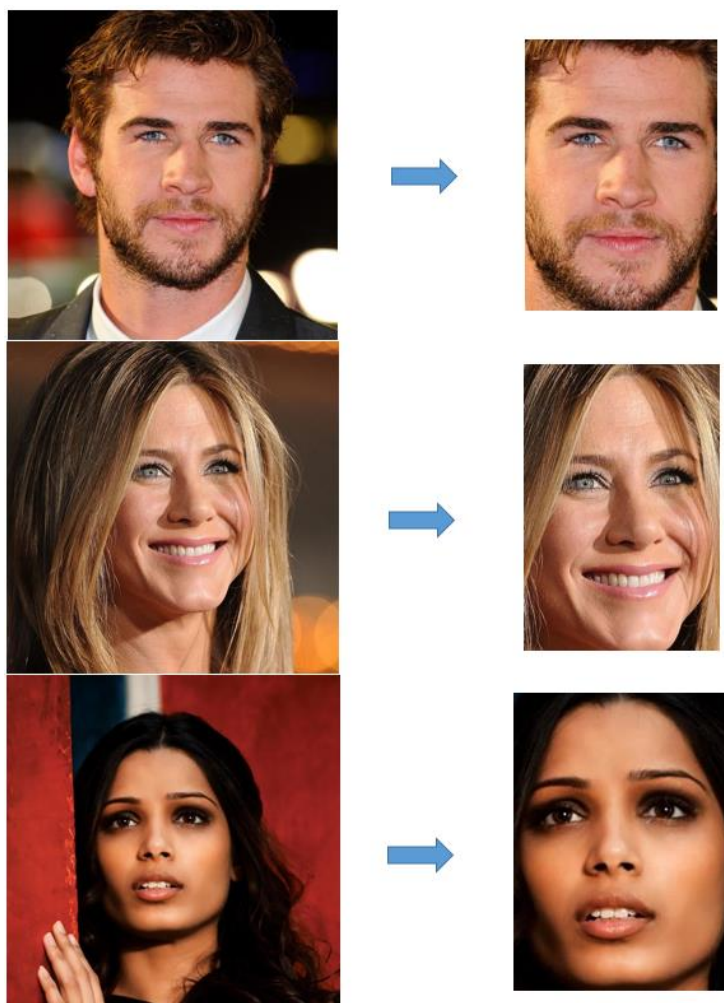
След тази обработка, броят на снимките драстично намалява от 523 051, до около 110 000.

Втората стъпка от обработката е да се изрежат и оставят само лицата от снимките. В оригиналните снимки хората често са снимани от кръста нагоре или дори в цял ръст. Да се отрежат и оставят само лицата има няколко положителни свойства:

- размера на данните намалява;
- могат да се използват по-малки невронни мрежи, тъй като няма да има нужда мрежите да се учат да разбират в коя част от снимката е лицето;
- изображенията, които ще се подават за предсказване, са на изрязани лица.

Логично е да се използва същия алгоритъм за намиране на лица, които ще се използва и във финалното приложение – този от точка 3.2.

Примери от резултата на изрязването са показани на *Фигура 22*.



Фигура 22. Преди и след изрязването на лицата от IMDB-WIKI базата от данни.

Алгоритъмът не успява да открие лица само на две от снимките (*Фигура 23*) и те не са добавени към финалните данни:

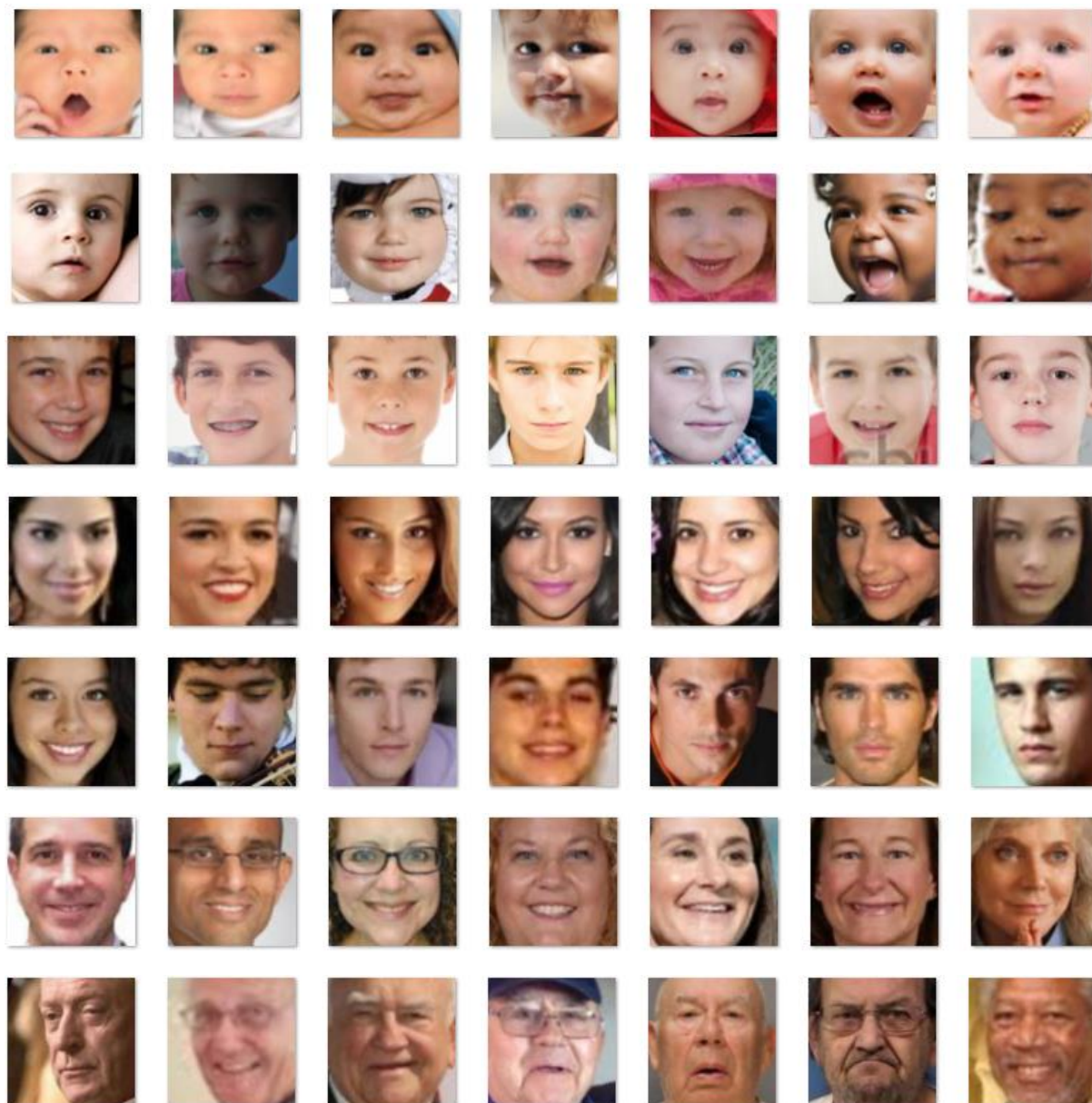


Фигура 23. Примерни за изображения, в които алгоритъма за откриване на лица не е успял да намери лице.

Както се вижда от графиката на *Фигура 21*, мнозинството от лица са между 20 и 50 годишни, а снимки на лица под 10 години почти няма. Това се дължи главно на факта, че по-голямата част от снимките са на известни актьори. Броят на лицата над 80 години също е доста нисък. Затова е прибавена още една база от данни – UKTFace.

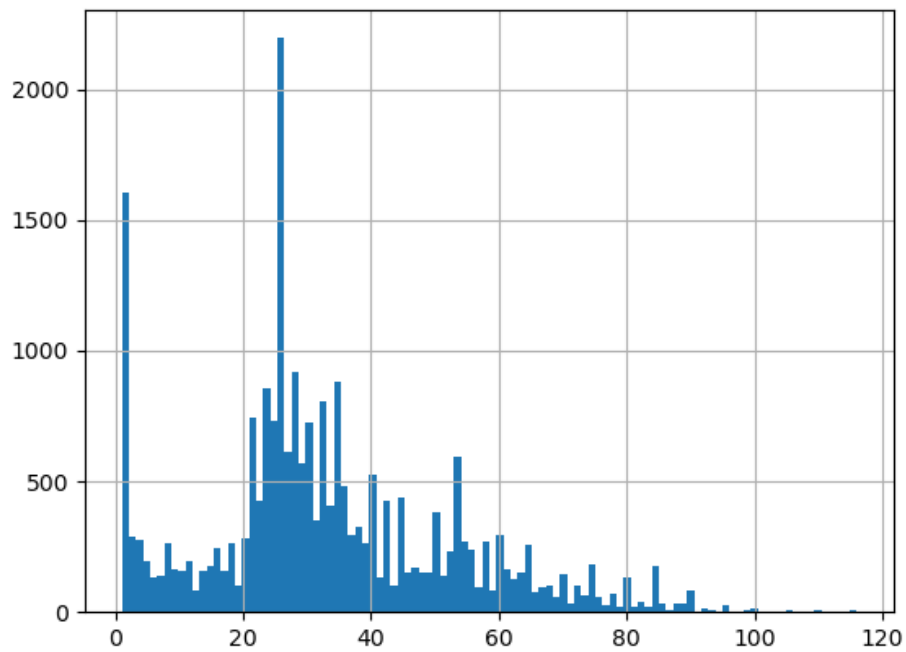
4.3 UKTFace

UKTFace [28] е голяма лицева база от данни с голям обхват на възрасти – от 1 до 116 години. Състои се от над 20 000 лицеви снимки, маркирани с възраст, пол и етническа принадлежност. За удобство, авторите ѝ предоставят версия с изрязани и подравнени лица (Фигура 24).



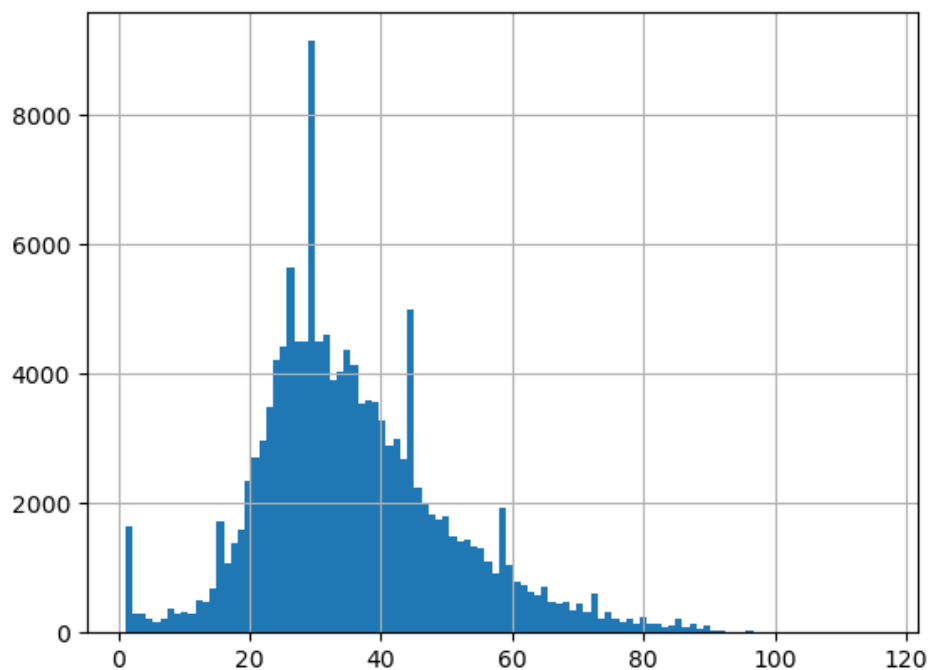
Фигура 24. Примерни снимки от UKTFace базата.

Както и при IMDB-WIKI, и тук мнозинството от снимки е на хора около 30-годишна възраст (Фигура 25).



Фигура 25. Разпределение на възрастта при UKTFace базата от данни.

Хистограмата след обединението на двете бази е представена на *Фигура 26*.



Фигура 26. Разпределение на възрастта при обединението на IMDB-WIKI и UKTFace.

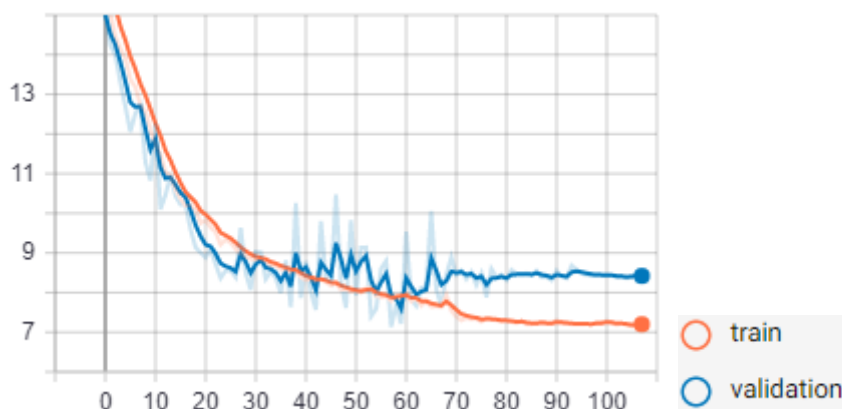
Метаданните за IMDB-WIKI базата се предоставени в *.mat* (MATLAB [29]) формат, а тези от UKTFace базата се съдържат в имената на всяко от изображенията. Например: **20**_1_0_20170117141358391.jpg.chip.jpg означава, че лицето е на 20

години и е с пол 1, т.е. женски за UKTFace. Тук има разминаване с IMDB-WIKI, тъй като там с 1 се означава мъжки пол.

За улеснение на по-нататъшната работа, данните са обработени и записани в csv формат. Съответно пола от двете бази е уеднаквен до 0 за женски и 1 за мъжки. Във финалния формат на данните са включени само 3 колони – пътят до изображението, възрастта и полът.

4.4 Допълнителна обработка след обединяване на данните

Първото „успешно“ обучение на модел за разпознаване на възраст е постигнато с регресионен модел, обучен само върху UKTFace данните. Mean Absolute Error грешката е сведена до 7.13 при епоха 58. Прогресът на обучението е показан на *Фигура 27*.



Фигура 27. Обучение на регресионен модел за разпознаване на възраст върху UKTFace данни.

На практика обаче почти всички хора, с които е експериментирано, са разпознати на възраст около 30 години (дори деца с реална възраст 2 години и хора на над 50 години). Причината за това е, че както се вижда на хистограмата от *Фигура 25*, мнозинството от хората на снимките са на тази възраст. Дори след обединението на двете бази, повечето от тестовите примери пак са в тази възрастова група (*Фигура 26*).

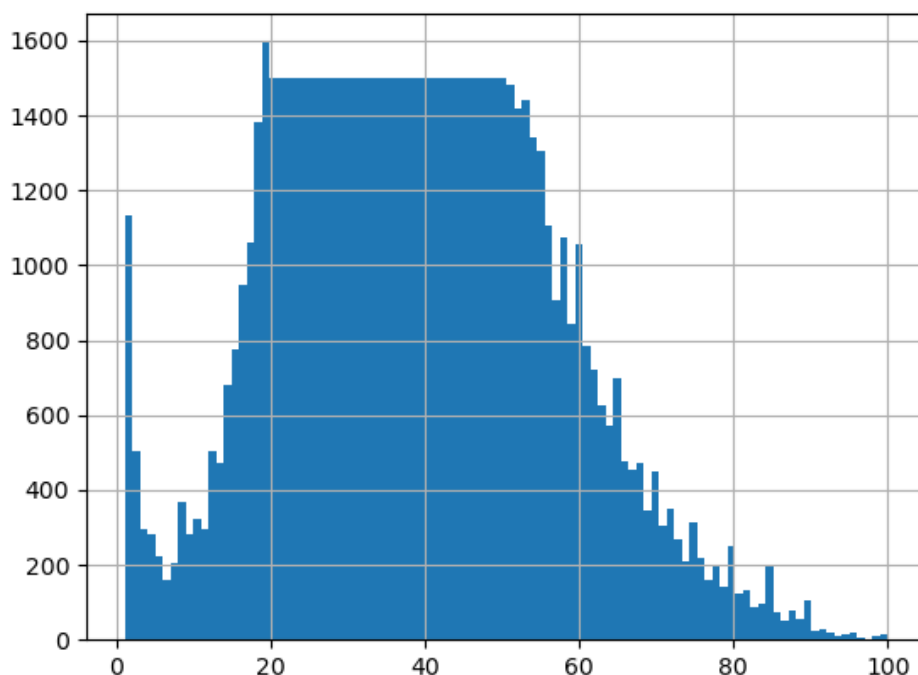
Затова следващата предприета стъпка е да се премахнат част от примерите, така че да бъдат по-равномерно разпределени. Както вече споменахме, а и както се вижда на *Фигура 28*, най-голямата част от примери са между 20 и 50 години.

1: 1133	11: 294	21: 2707	31: 4495	41: 2901	51: 1482	61: 783	71: 304	81: 125	91: 24
2: 502	12: 502	22: 2959	32: 4596	42: 2980	52: 1417	62: 723	72: 349	82: 131	92: 27
3: 295	13: 472	23: 3490	33: 3905	43: 2687	53: 1441	63: 628	73: 266	83: 89	93: 17
4: 282	14: 682	24: 4225	34: 4039	44: 2358	54: 1343	64: 573	74: 208	84: 97	94: 9
5: 225	15: 773	25: 4415	35: 4377	45: 2628	55: 1304	65: 698	75: 313	85: 201	95: 15
6: 159	16: 946	26: 5648	36: 4134	46: 2247	56: 1107	66: 475	76: 219	86: 75	96: 19
7: 206	17: 1061	27: 4496	37: 3548	47: 1969	57: 906	67: 453	77: 160	87: 52	97: 4
8: 369	18: 1381	28: 4511	38: 3602	48: 1830	58: 1075	68: 474	78: 202	88: 77	98: 2
9: 283	19: 1594	29: 4577	39: 3558	49: 1734	59: 842	69: 346	79: 143	89: 54	99: 12
10: 323	20: 2347	30: 4568	40: 3284	50: 1797	60: 1054	70: 450	80: 251	90: 107	100: 16

Фигура 28. Брой снимки за всяка възраст при обединените IMDB-WIKI и UKTFace бази от данни.

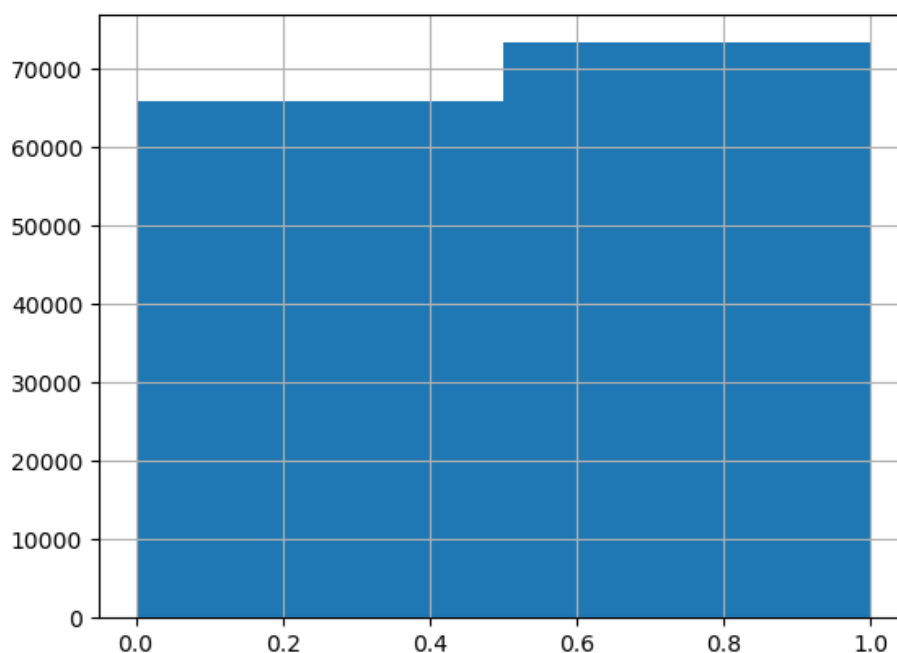
Средно има 1392 снимки за всяка възраст. Затова възрастите между 20 и 50 години бяха ограничени до 1500 снимки за възраст. Тъй като е невъзможно (или поне ще отнеме изключително много време) тези снимки да се изберат на ръка, взети са 1500-те най-големи по памет снимки за всяка от тези 30 възрастови групи. Интуицията е, че по-големите снимки са с по-добро качество и ще допринесат повече за обучаването на модела.

Резултатът от тази операция може да се види на *Фигура 29*, което е и базата от данни, използвана за трениране на модела за разпознаване на възраст.



Фигура 29. Разпределение на възрастовите групи след ограничаването на класовете между 20 и 50 години.

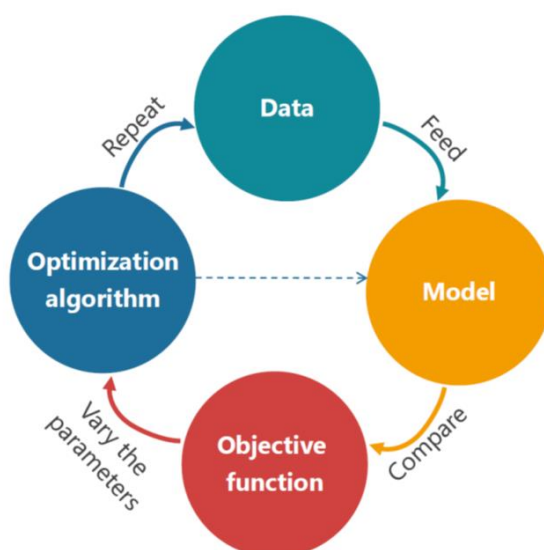
За обучаване на модела за разпознаване на пол е използвана базата преди „подрязването“. Снимките спрямо пол са почти равномерно разпределени – *Фигура 30*.



Фигура 30. Разпределение на пола при IMDB-WIKI и UKTFace обединените бази.

5. Предсказване на възраст от изображение на лице

Базовата логика за трениране на един алгоритъм за дълбоко обучение се състои от четири стъпки – подготовка на данни, избор на модел, избор на обективна функция (objective function) и избор на оптимизационен алгоритъм (Фигура 31).



Фигура 31. Процес на разработване на алгоритъм за дълбоко обучение. [30]

След като вече имаме подготвени данните за обучение, можем да преминем към избора на модел за решаване на втората от поставените в точка 1.3 задачи – тази за предсказване на възраст от изображение на лице.

5.1 Избор на модел

Първоначалната идея беше да се вземе някоя от архитектурите от точка 2.4 и да се тренира върху подготвената база от данни от точка 4. Преди това обаче бяха направени тестове с два вече готови модела, обучени за разпознаване на възраст.

Единият модел е представен с IMDB-WIKI базата от данни – DEX: Deep Expectation of apparent age from a single image [31].

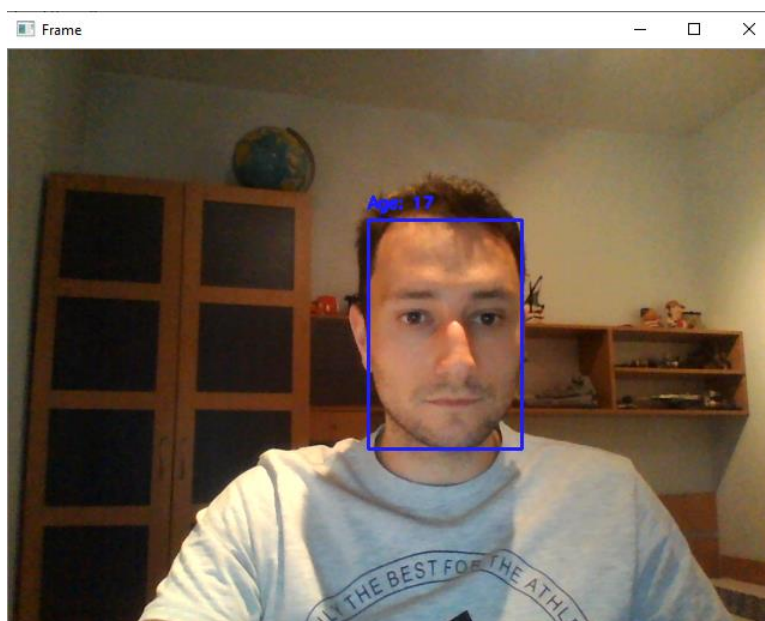
Вторият модел е базиран на VGG-16 архитектурата [32] и използва трансферирано обучение (transfer learning). Предварително е трениран върху данните от ImageNet [33], а изходните слоеве са допълнително тренирани върху IMDB-WIKI базата от данни.

За съжаление и на двата модела им отнемаше повече от 0.4 секунди за едно единствено предсказание (*Фигура 32*), което ги прави неприложими за система, която трябва да работи в реално време.

```
Found face [315 264 433 438] with confidence 0.973556  
time to predict age: 0.4077732563018799  
time to predict gender: 0.03397512435913086  
Gender: male
```

Фигура 32. Време за предсказване на DEX модела.

А както се вижда на *Фигура 33*, DEX моделът дори не показва обещаващи резултати при експериментите.



Фигура 33. Предсказване на възраст с DEX модела. Предсказана възраст – 17. Реална възраст – 27.

Затова е използвана стандартна архитектура на конволюционна невронна мрежа, описана в точка 2.4. И по-точно – редуване на конволюционни със свързващи слоеве и завършване с напълно-свързани слоеве. След известна проба и грешка, финалният модел е представен на *Фигура 34*.

```
tf.keras.models.Sequential([
    Input(shape=image_shape),
    Conv2D(64, 3, activation='relu'),
    Conv2D(64, 3, activation='relu'),
    MaxPool2D(2),
    Dropout(0.3),
    Conv2D(128, 3, activation='relu'),
    Conv2D(128, 3, activation='relu'),
    MaxPool2D(2),
    Dropout(0.3),
    Conv2D(196, 3, activation='relu'),
    Dropout(0.3),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(1)
])
```

Фигура 34. Модела, използван за предсказване на възраст. Общ брой параметри – 2 366 725.

5.2 Обучение

Тъй като за предсказване на възрастта е избран регресионен модел, използваната обективна функция за грешката е MAE или Mean Absolute Error. Тя се представя със следната формула:

$$MAE(X, h) = \frac{1}{m} \sum_{i=1}^m |h(x^{(i)}) - y^{(i)}|$$

Където:

- m е броят на данните, с които тестваме грешката;
- $x^{(i)}$ е i -тото изображение от данните, с които тестваме;
- X са всички данни, с които тестваме;
- h е функцията, която предсказва. Още се нарича хипотеза;
- $y^{(i)}$ е реалната стойност на свойството на $x^{(i)}$, което предсказваме.

Използваният оптимизационен алгоритъм е Adam [34].

Активационната функция, използвана за всички конволюционни слоеве и напълно свързания слой преди изходния, е ReLU (*Фигура 9*).

Преди да се подаде на модела, всяко изображение се конвертира до черно-бяло и се оразмерява до 60×80 . Премахването на цвета се прави с цел да се намали броят на параметрите и за да се игнорира цветът на кожата като фактор. А размерът

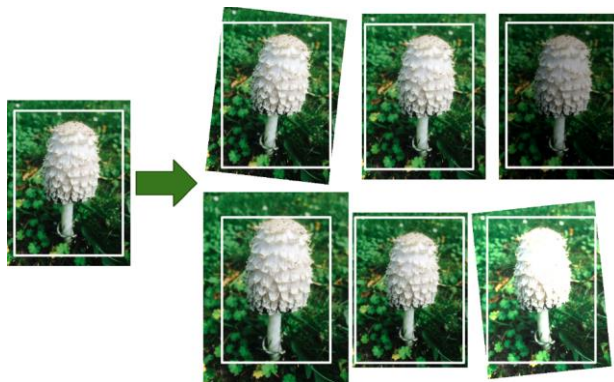
60 × 80 е избран, тъй като е стандартно отношение на портретна снимка, а след изрязването, отношението ширина-дължина на изображенията е подобно (Фигура 22).

Изображенията се подават на порции от по 128, защото няма как всички да бъдат заредени едновременно в паметта.

Данните са разделени на 3 части – 80% за обучение, 10% за валидация и 10% за тестване.

След ограничаване броя на изображенията с възраст между 20 и 50 години до 1500 за възрастова група (Фигура 29), броят на данните е значително намален до 79 124. Но по време на обучението се използва т.н. техника за изменение на данни (Data Augmentation) върху данните, отделени за обучение.

При изменението на данни изкуствено се увеличава броят на данните за обучение като се генерират реалистични инварианти за всяко от изображенията (Фигура 35). Измененията трябва да са толкова реални, че човек да не може да познае дали дадено изображение е изменено, или не. Пример за изменения са леки измествания, завъртания, приближения и оразмерявания. Колкото и да са малки и дори незабележими на пръв поглед за човек, за компютъра това са съвсем нови примери.

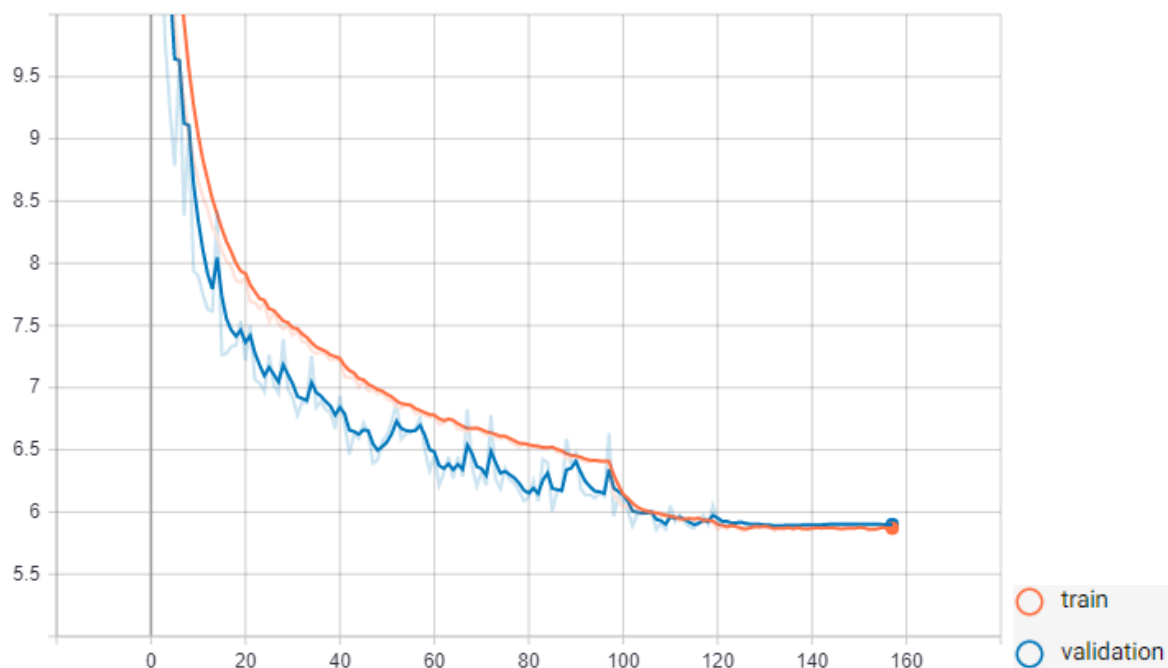


Фигура 35. Изменение на данните – генериране на нови обекти за трениране от вече съществуващи. [7]

Моделът се тренира докато няма подобрене при валидацията с търпение от 50 епохи.

Първоначалната стъпка на обучение (learning rate) е 0.001, но при 12 последователни валидации без подобрене се намалява, като се умножава по 0.1. При всяка валидация с подобрене, теглата се записват във файл, с цел после да бъдат използвани за тестване и предсказване. Цялото обучение се следи с TensorBoard [35].

Моделът се тренира 16 ч, 49 мин и 11 сек на GPU NVIDIA GeForce GTX 760m и постига намаляване на грешката до **5.86** при 108-ма епоха (Фигура 36). Общ брой на епохите – 157.



Фигура 36. Обучение на модела за разпознаване на възраст.

След 96-та епоха има намаляне на стъпката на обучение.

След оценяване на модела с тестовите данни, резултата за грешката е **5.9652**.

Предсказването на едно изображение отнема около **0.03** секунди.

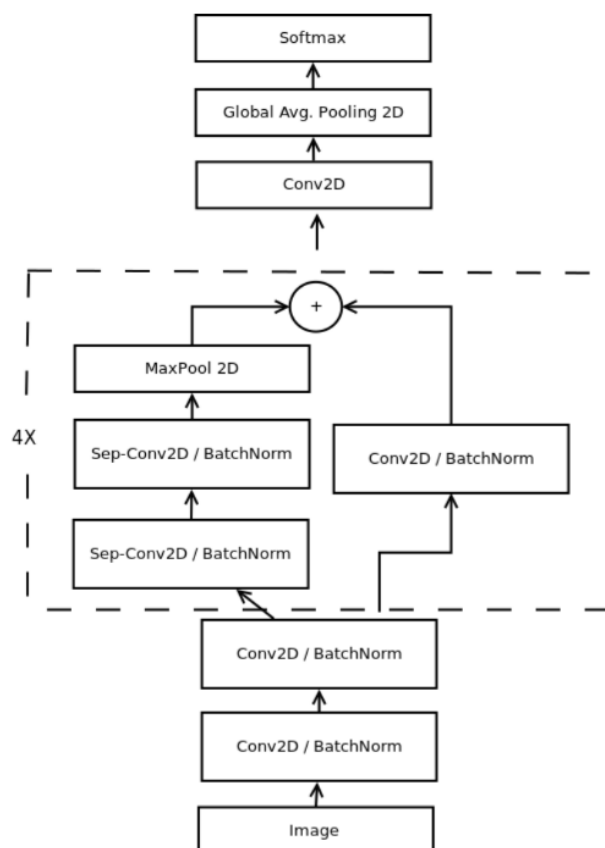
6. Предсказване на пол от изображение на лице

Последната от поставените в точка 1.3 задачи е задачата за предсказване на пол от изображение на лице. Тя се разглежда като класификационна задача с два класа (пола).

6.1 mini-Xception

Първият направен опит е с олекотена версия на архитектурата Xception [14]. В публикацията си авторите я наричат mini-Xception [36], а архитектурата ѝ е показана на *Фигура 37*.

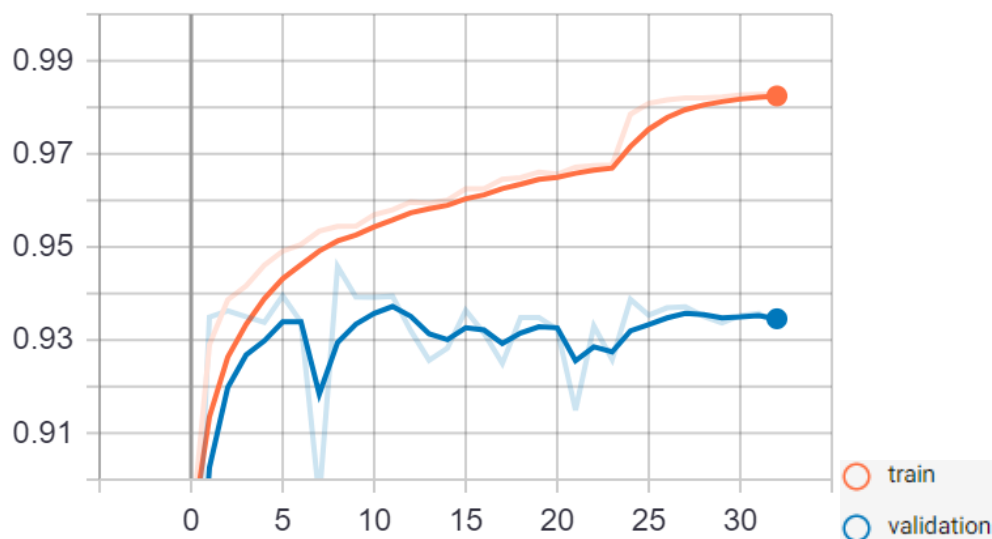
Още при първата епоха моделът успява да достигне точност от над 90%, а най-добрата си точност достига едва на деветата епоха и то след само 40 минути обучение. Както обаче се вижда и от графиката на *Фигура 38*, след това валидацията започва много да се разминава с обучението и няма по-нататъшно подобрене.



Фигура 37. Архитектура mini-Xception.

След оценка с тестовите данни, моделът показва точност 94.2% със загуба 0.18.

Първоначалната стъпка на обучение е 0.001, като след 12 епохи без подобрение, тя се намалява, като се умножава по 0.1. Това се е случило след 24-тата епоха като е подобрило тренирането, но не е оказало влияние на валидацията. От графиката се вижда, че още след около петата епоха моделът е започнал да се нагажда спрямо тренировъчните данни (overfitting).



Фигура 38. Обучение на модела mini-Xception.

6.2 Традиционна конволюционна невронна мрежа

По-късно, когато беше разработена конволюционната мрежа, използвана за разпознаване на възраст от точка 5.1, тя е приложена и върху задачата за разпознаване на пол – *Фигура 39*.

Разликата е, че вече се решава класификационна задача и последният регресионен слой е заместен със слой, който произвежда два изхода и използва активиращата функция softmax (виж *Фигура 9*).

Използваната обективна функция е Categorical Cross-Entropy loss, а оптимизационния алгоритъм е отново Adam [34]. Categorical Cross-Entropy loss е softmax активация плюс Cross-Entropy loss функция. Cross-Entropy loss функцията се представя със следната формула:

$$CE = - \sum_i^C t_i \log(s_i)$$

Където t_i е реалната стойност на разглеждания пример, а s_i е предположената вероятност примерът да принадлежи към клас i от набора от класове C .

Categorical Cross-Entropy loss се смята като s_i от горната формула се замести с $\sigma(s)_i$, т.е. softmax функцията, представена на *Фигура 9*.

Моделът се тренира 14 ч, 52 мин и 8 сек, като не е изчакан да приключи поради липса на подобрение след зададения брой епохи, а обучението е прекратено ръчно, тъй като последните 3 часа подобрението е в рамките на хилядни – *Фигура 40*.

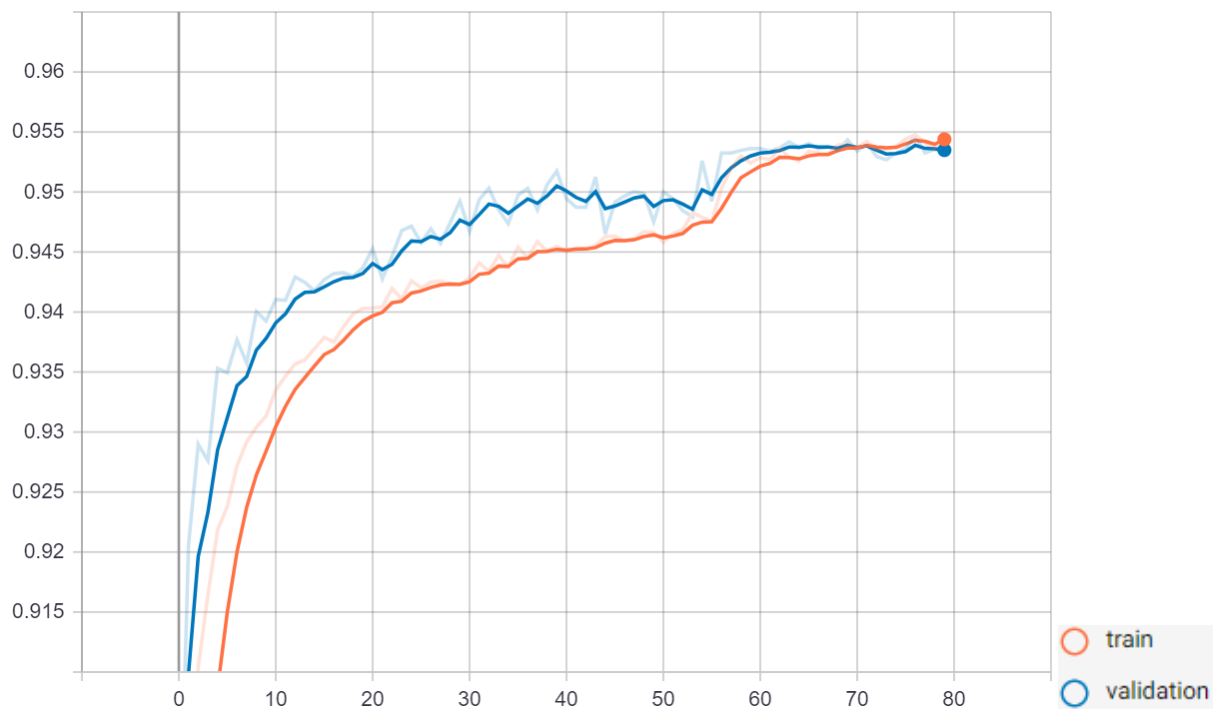
Резултатите след оценяване с тестовите данни са точност **95.92%** и грешка **0.12**. Почти 2% подобрение над mini-Xception модела.

За предсказване на едно изображение моделът отнема около **0.03** секунди.

Това е и моделът, който се използва за разпознаване на пол във финалното приложение.

```
tf.keras.models.Sequential([
    Input(shape=image_shape),
    Conv2D(64, 3, activation='relu'),
    Conv2D(64, 3, activation='relu'),
    MaxPool2D(2),
    Dropout(0.3),
    Conv2D(128, 3, activation='relu'),
    Conv2D(128, 3, activation='relu'),
    MaxPool2D(2),
    Dropout(0.3),
    Conv2D(196, 3, activation='relu'),
    Dropout(0.3),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(2, activation='softmax'),
    # Dense(1) // for age regression
])
```

Фигура 39. Структура на конволюционната невронна мрежа, използвана за разпознаване на пол.



Фигура 40. Обучение на конволюционната невронна мрежа за разпознаване на пол.

7. Използван софтуер и работа на приложението

7.1 Използван софтуер и хардуер

Приложението е разработено с помощта на програмния език Python 3.7 [37]. За обработване на базите от данни е използвана библиотеката Pandas [38].

Моделът за намиране на лица в изображение и придружаващите го тегла са предоставени от библиотеката OpenCV [18].

За създаване и трениране на моделите за разпознаване на възраст и пол е използвана библиотеката TensorFlow 2.0 [39].

Моделите са тренирани върху видео карта на машина със следната конфигурация:

Процесор: Intel i7-4702 MQ @ 2.2 GHz

Оперативна памет: 8GB

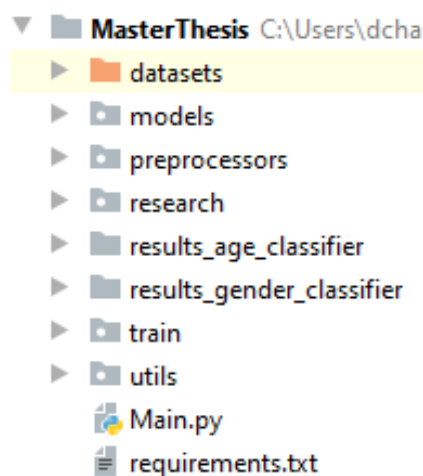
Видео карта: NVIDIA GeForce GTX 760m със 768 CUDA ядра и 2GB GDDR5 памет.

Поради това, че видео картата е с Compute Capability 3.0, а TensorFlow след версия 1.14 поддържа карти само с Compute Capability поне 3.5 (заради поддръжката на множество GPU-та), се наложи TensorFlow да бъде компилиран от код, като се промени да поддържа Compute Capability 3.0.

2GB се оказаха изключително малко и не успяха да поберат обучението на никой от моделите от точка 2.4, с които бяха правени опити.

7.2 Файлова структура и стартиране на приложението

Файловата структура на проекта е представена на *Фигура 41*.



Фигура 41. Файлова структура на проекта.

Където:

- *datasets* – съдържа базите от данни със снимки;
- *models* – съдържа използваните модели;
- *preprocessors* – съдържа различни скриптове, които са използвани за предварителна обработка на данните;
- *research* – съдържа различни скриптове, които са използвани по време на проучването и разработването на проекта;
- *results_age_classifier* и *results_gender_classifier* съдържат резултатите от различните опити с модели, които са правени;
- *train* – съдържа скриптовете, използвани за обучаване на моделите;
- *utils* – съдържа имплементацията на моделите, които са тествани и callback методите, които са ползвани;
- *Main.py* е стартовата точка на приложението и в действителност съдържа цялата логика;
- *requirements.txt* – помощен файл, който указва какви библиотеки са нужни за работата на приложението.

За да се инсталират нужните библиотеки, е достатъчно да се изпълни следната команда:

```
$ pip install -r requirements.txt
```

Това ще инсталира библиотеките:

- TensorFlow 2.0
- numpy 1.16.4
- opencv-python 4.1.0.25
- imutils 0.5.2

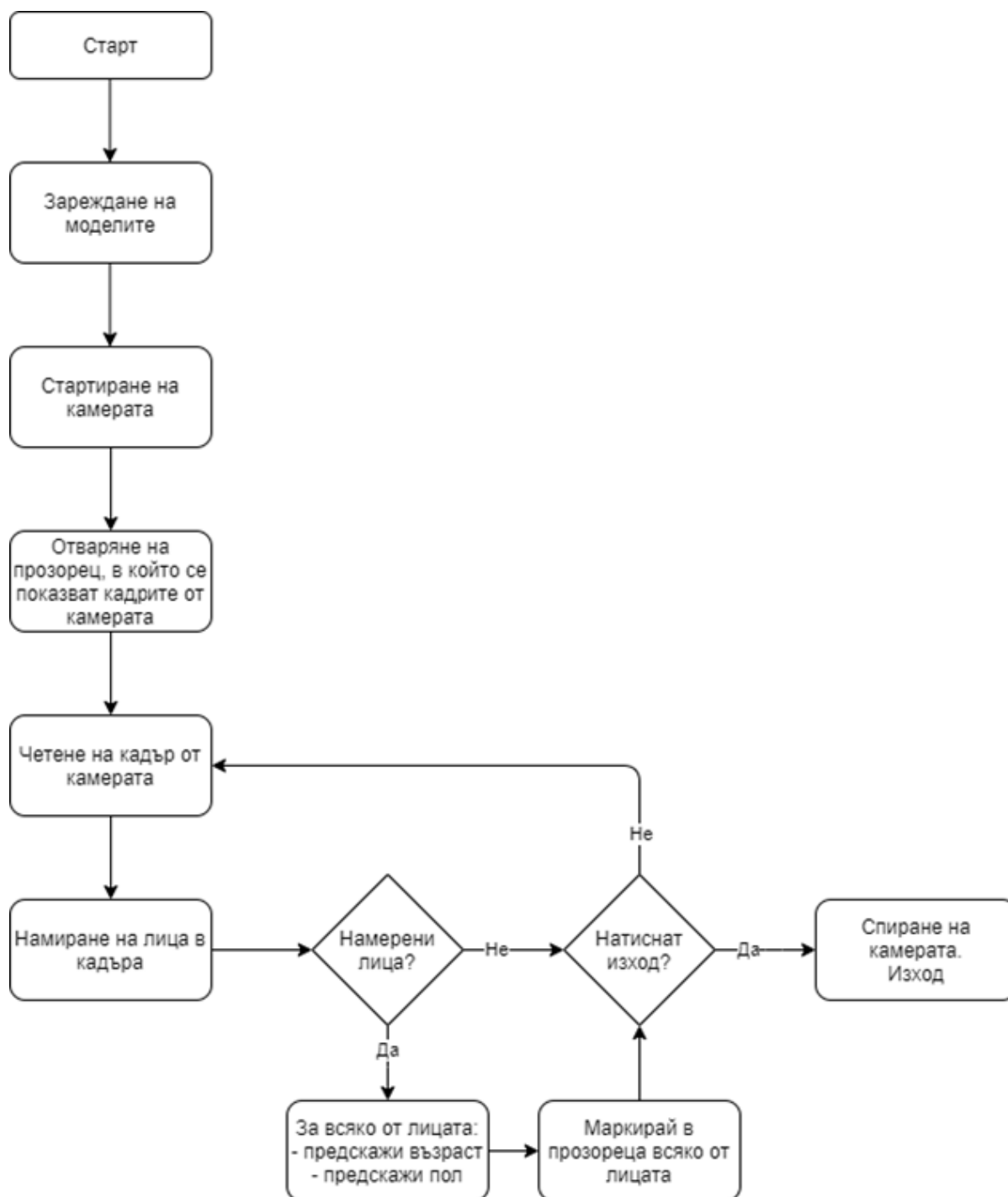
Това са библиотеките, нужни за работа на завършеното приложение. По време на разработката му са използвани и библиотеки като *pandas*, *matplotlib*, *scikit-learn*, *graphviz* и други.

Единственото нещо, което трябва да е предварително инсталирано на системата е Python 3.7.

За да се стартира приложението, трябва да се изпълни файла *Main.py*:

```
$ python ./Main.py
```

Процеса на работа на приложението е описан на *Фигура 42*.



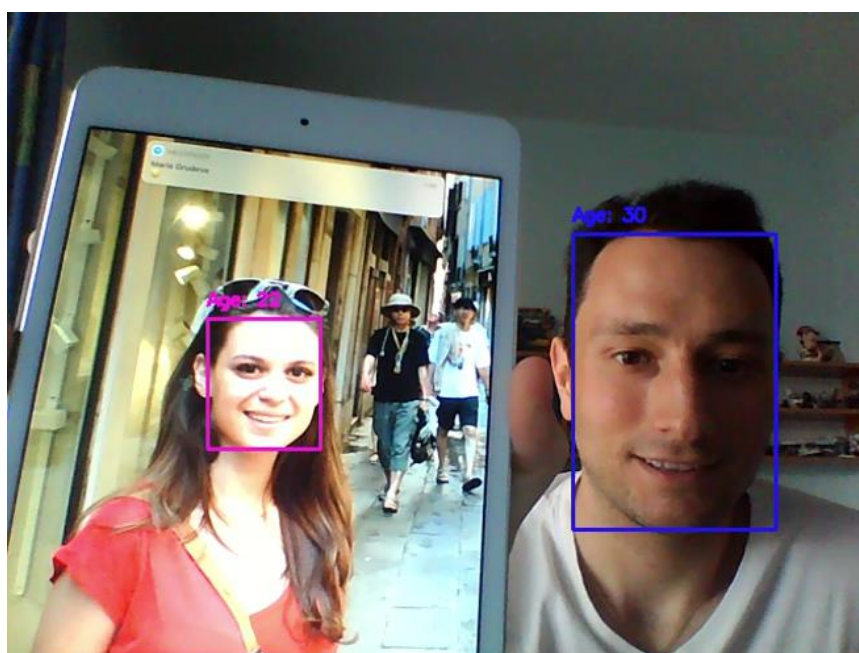
Фигура 42. Процес на работа на приложението.

Маркирането се изразява в това да се огради лицето с правоъгълна кутия, като полът е изразен чрез цвета на кутията – син за мъжки и розово-лилав за женски. Предположената възраст е изписана над горния ляв край на кутията.

8. Проведени експерименти

В тази секция ще разгледаме част от проведените експерименти със завършеното приложение.

Експеримент 1 е с две лица, едното от които е пред камерата, а другото е показано на снимка. Полът на двете лица и годините на лицето от женски пол са правилно разпознати, но при възрастта на лицето от мъжки пол има разминаване от три години. От този експеримент също така се вижда и, че алгоритъмът за намиране на лица наистина работи с различна големина на лицата.



Експеримент 1. Мъж на 27 и жена на 22 години.



Реална възраст – 22
Предсказана – 22

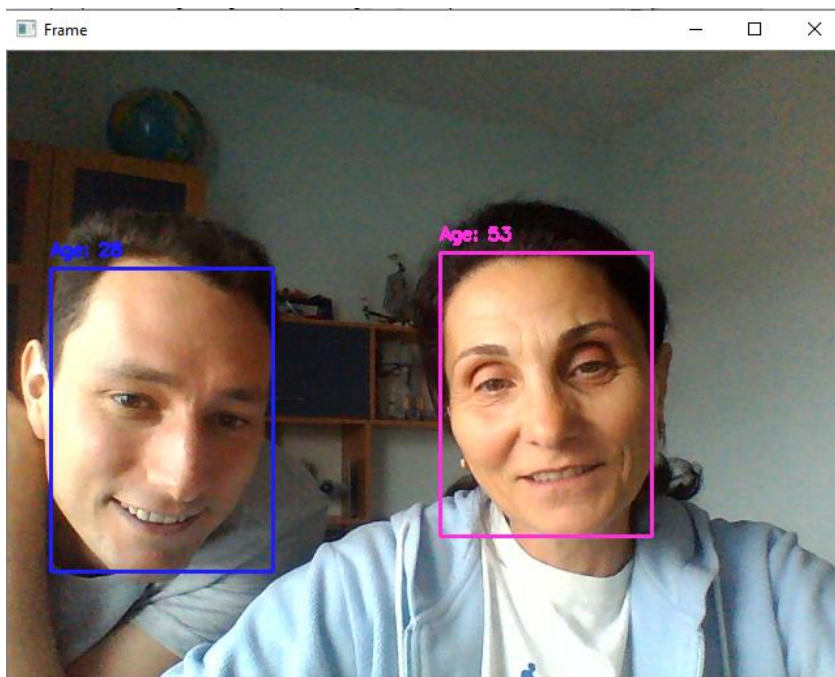


Реална възраст – 27
Предсказана – 30

Експеримент 2 е с две лица пред камерата. Едното е от мъжки пол на действителна възраст 27, а другото – от женски пол с действителна възраст 55. Полът и на двете лица е правилно разпознат, но при възрастта има разминавания, съответно с една и две години.

Експеримент 3 е със снимка на дете на 2 години. Възрастта е правилно разпозната, но полът е грешен. Снимката е на момче, а моделът го е класифицирал като момиче. От проведените експерименти се установи, че моделът за разпознаване на пол често дава грешни резултати при тестове с малки деца.

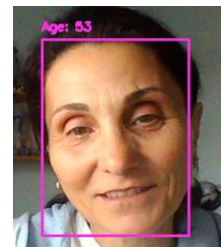
Експеримент 4 е с три лица, две от които са показани на снимка. Полът на трите лица и възрастта на лицето от мъжки пол са правилно разпознати, но при другите две лица има разминаване. Жената с действителна възраст 55 е разпозната като 45, а двугодишното дете е разпознато като тригодишно.



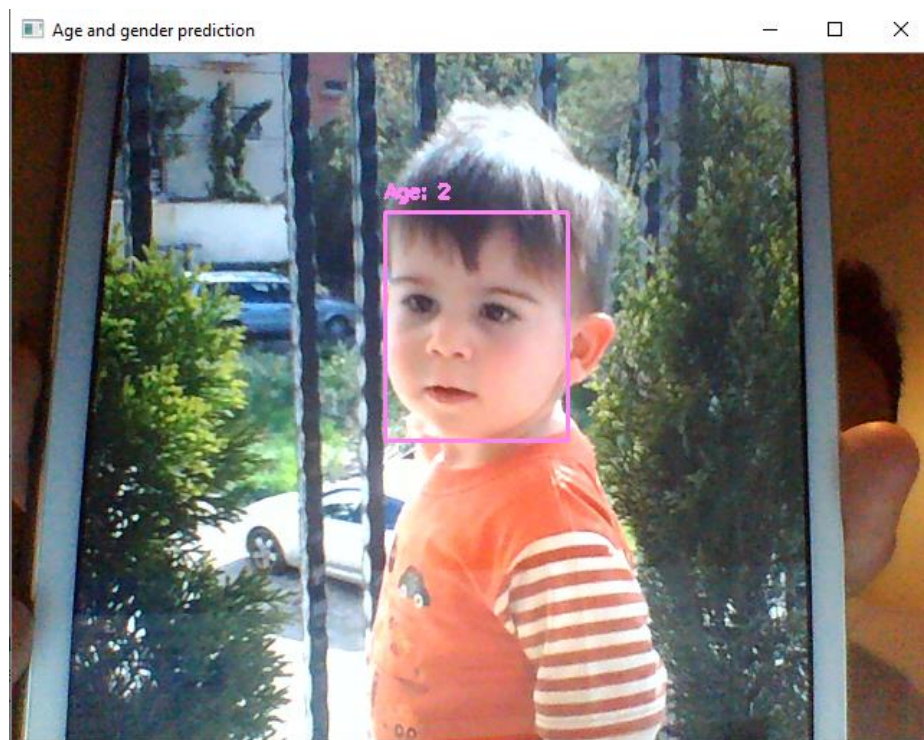
Експеримент 2. Мъж на 27 и жена на 55 години.



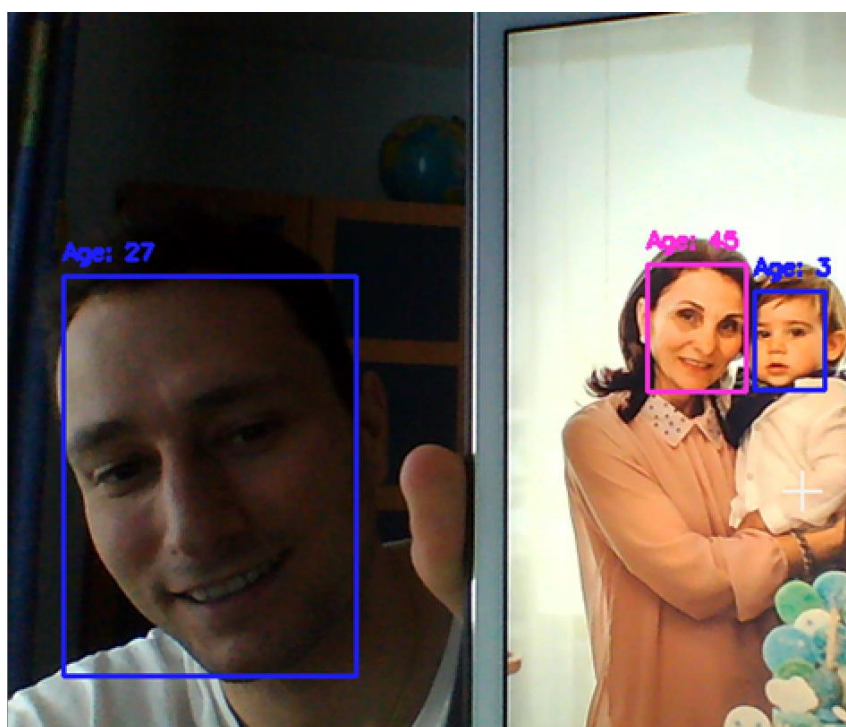
Реална възраст – 27
Предсказана – 28



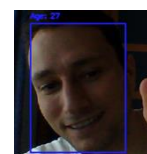
Реална възраст – 55
Предсказана – 53



Експеримент 3. Момче на 2 години.



Експеримент 4. Мъж на 27, жена на 55 и момче на 2 години.



Реална възраст – 27
Предсказана – 27



Реална възраст – 55
Предсказана – 45



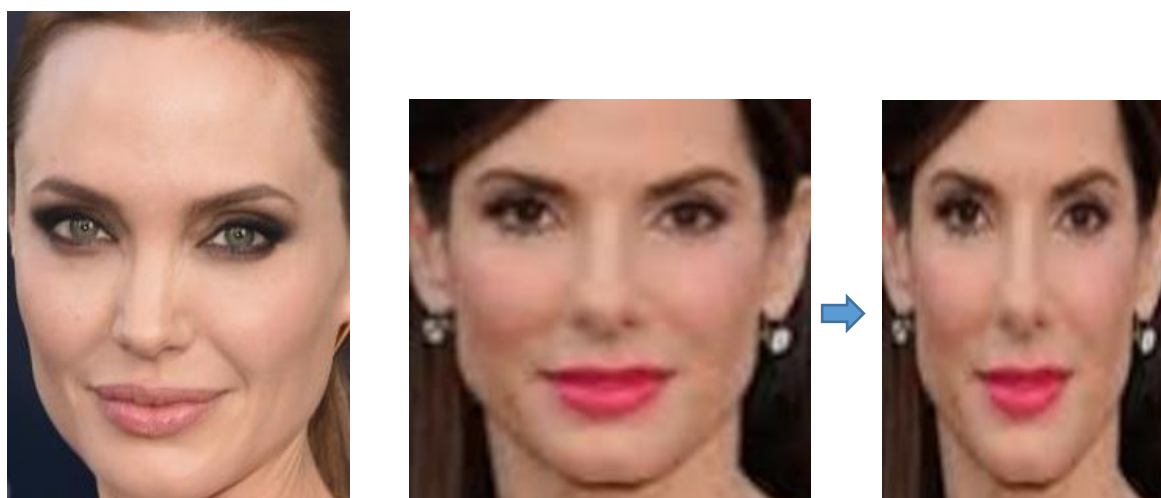
Реална възраст – 2
Предсказана – 3

Разпознаването на действителната възраст е трудна задача дори и за хората. От проведените експерименти се установи, че макар винаги да е близо, моделът често греши при разпознаване на действителната възраст, но се справя доста добре при разпознаване на *видимата* възраст.

9. Бъдещо развитие

Има няколко неща, които биха могли да се предприемат като следващи стъпки:

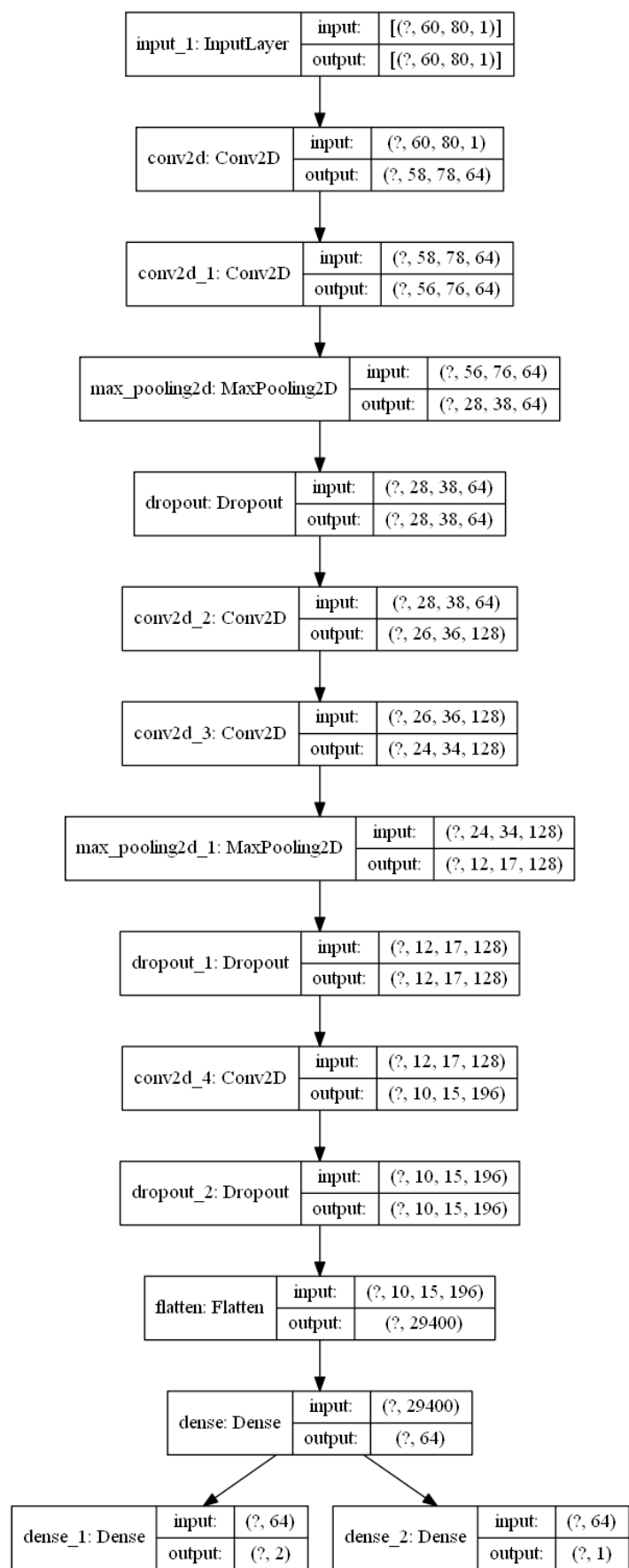
- Тъй като UKTFace данните са вече на изрязани лица, те не са обработвани по никакъв допълнителен начин. Снимките там, обаче, са квадратни, а снимките от IMDB-WIKI след обработката са правоъгълни с отношение от около 3:4. Разработените модели очакват снимките да са предварително оразмерени до 60×80 , което не би променило много отношението в обработените снимки от IMDB-WIKI, но на тези от UKTFace – да (Фигура 43). Затова, едно от нещата, които биха могли да бъдат направени, е снимките от UKTFace да минат през същата предварителна обработка, както тези от IMDB-WIKI.



Фигура 43. Ляво – примерна снимка от IMDB-WIKI след обработката. По средата – примерна снимка от UKTFace. Дясно – снимка от UKTFace след оразмеряване.

- Обединяване на двете предсказания под един модел. Това може да бъде постигнато като използваната невронна мрежа се преобразува в т.н. двуглава невронна мрежа. Прототип на такъв модел е представен на Фигура 44.
- Въпреки предварителната обработка на данните, множеството примери пак са между 20 и 50 години. Може би биха могли да се добавят още данни с цел възрастта да се разпредели по-равномерно.
- Тъй като машината, на която е разработено приложението е ограничена откъм ресурси, не са използвани по-дълбоки модели. Може би такива модели биха могли да бъдат тренирани върху Google Cloud [40].

Приложението може да бъде разширено с функционалност за разпознаване на емоции или за да бъдем по-актуални, разпознаване дали лицето е с маска, или не.



Фигура 44. Прототип на конволюционна невронна мрежа с два изхода, предназначена за обединяване на двете задачи в един модел.

10. Заключение

В настоящия документ беше представена система за разпознаване на пол и възраст на лица от видео в реално време.

За постигане на целта са използвани конволюционни невронни мрежи, програмният език Python 3.7 и библиотеката TensorFlow 2.0.

Отделните модели постигат следните резултати:

- MAE грешка **5.9652** при тестване на модела за разпознаване на възраст;
- точност **95.92%** и грешка **0.12** при тестване на модела за разпознаване на пол.

И на двата модела им отнема около **0.03** секунди за едно предсказание, което не забавя осезаемо работата на приложението и то може да работи в реално време.

С това можем да заключим, че целите, поставени в точка 1.3 са постигнати.

В процеса на разработка се създаде и нова лицева база от данни, която обединява IMDB-WIKI и UKTFace базите, но непълните или грешни данни от IMDB-WIKI са премахнати, а самите снимки са изрязани така, че в тях да се съдържат само лицата.

Литература

- [1] B. Davis и O. Avci, „Simplified Vibration Response Prediction for Slender Monumental Stairs,“ в *Structures Congress*, Boston, Massachusetts, United States, 2014.
- [2] „Mass surveillance in China,“ [Онлайн]. Available: https://en.wikipedia.org/wiki/Mass_surveillance_in_China.
- [3] O. Sendik и Y. Keller, *DeepAge: Deep Learning of face-based age estimation*, 2019.
- [4] A. Dhomne, R. Kumar и V. Bhan, *Gender Recognition Through Face Using Deep Learning*, 2018.
- [5] D. H. Hubel, *SINGLE UNIT ACTIVITY IN STRIATE CORTEX OF UNRESTRAINED CATS*, Washington, 1958.
- [6] D. H. Hubel и T. N. Wiesel, *RECEPTIVE FIELDS OF SINGLE NEURONES IN THE CAT'S STRIATE CORTEX*, Baltimore, Maryland, 1959.
- [7] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, Canada: O'Reilly, 2019.
- [8] K. Fukushima, *Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position*, Kinuta, Tokyo, 1980.
- [9] Y. LeCunn, *Gradient-Based Learning Applied to Document Recognition*, 1998.
- [10] A. Krizhevsky, *ImageNet Classification with Deep Convolutional Neural Networks*, 2012.
- [11] K. Simonyan и A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*.
- [12] C. Szegedy, *Going Deeper with Convolutions*, 2014.
- [13] K. He и X. Zhang, *Deep Residual Learning for Image Recognition*, 2015.
- [14] F. Chollet, *Xception: Deep Learning with Depthwise Separable Convolutions*, 2016 .
- [15] „Keras,“ [Онлайн]. Available: <https://keras.io/>.
- [16] P. Viola и M. Jones, „Rapid Object Detection using a Boosted Cascade of Simple Features,“ в *Conference on Computer Vision and Pattern Recognition*, 2001.
- [17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu и A. Berg, *SSD: Single Shot MultiBox Detector*, 2016.

- [18] „OpenCV,” [Онлайн]. Available: <https://opencv.org/>.
- [19] S. Mallick, „Histogram of Oriented Gradients,” 2016. [Онлайн]. Available: <https://www.learnopencv.com/histogram-of-oriented-gradients/>.
- [20] „Dlib,” [Онлайн]. Available: <http://dlib.net/>.
- [21] D. King , „Easily Create High Quality Object Detectors with Deep Learning,” 2016. [Онлайн]. Available: <http://blog.dlib.net/2016/10/easily-create-high-quality-object.html>.
- [22] „Labeled Faces in the Wild,” [Онлайн]. Available: <http://vis-www.cs.umass.edu/lfw/>.
- [23] D. King, „frontal_face_detector.h,” dlib, [Онлайн]. Available: https://github.com/davisking/dlib/blob/master/dlib/image_processing/frontal_face_detector.h.
- [24] D. King, *Max-Margin Object Detection*, 2015.
- [25] „IMDB-WIKI – 500k+ face images with age and gender labels,” 2015. [Онлайн]. Available: <https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/>.
- [26] „IMDB,” [Онлайн]. Available: <https://www.imdb.com/>.
- [27] „Wikipedia,” [Онлайн]. Available: <https://www.wikipedia.org/>.
- [28] „UKTFace Large Scale Face Dataset,” [Онлайн]. Available: <https://susanqq.github.io/UTKFace/>.
- [29] „MATLAB,” [Онлайн]. Available: <https://www.mathworks.com/products/matlab.html>.
- [30] „Deep Learning with TensorFlow 2.0 course,” 365 Data Science, [Онлайн]. Available: <https://365datascience.com/courses/deep-learning-with-tensorflow-2-0/>.
- [31] R. Rothe, R. Timofte и L. V. Gool, *DEX: Deep EXpectation of apparent age from a single image*.
- [32] S. I. Serengil, „Apparent Age and Gender Prediction in Keras,” 2019. [Онлайн]. Available: <https://sefiks.com/2019/02/13/apparent-age-and-gender-prediction-in-keras>.
- [33] „ImageNet,” [Онлайн]. Available: <http://www.image-net.org/>.
- [34] D. P. Kingma и J. Ba, *Adam: A Method for Stochastic Optimization*, 2014.
- [35] „TensorBoard,” [Онлайн]. Available: <https://www.tensorflow.org/tensorboard>.
- [36] O. Arriaga, M. Valdenegro-Toro и P. Ploger, *Real-time Convolutional Neural Networks for emotion and gender classification*, 2019.

- [37] „Python,“ [Онлайн]. Available: <https://www.python.org/>.
- [38] „Pandas,“ [Онлайн]. Available: <https://pandas.pydata.org/>.
- [39] „TensorFlow,“ [Онлайн]. Available: <https://www.tensorflow.org/>.
- [40] „Google Cloud,“ [Онлайн]. Available: <https://cloud.google.com/>.