

Факултет по математика и информатика

Катедра „Компютърна информатика“

Дипломна работа

На тема

Разпознаване на емоции на лица

**Ръководител**

доц. В. Симеонова

**Дипломант**

Щилиян Александров Узунов

ФН: 24676

Съдържание

[1 Увод 3](#_Toc40729450)

[1.1 Дефиниция за „Разпознаване на емоции“ 3](#_Toc40729451)

[1.2 FACS Дефиниция 4](#_Toc40729452)

[1.3 Цел на дипломната работа 7](#_Toc40729453)

[2 Преглед на използваните научни методи 8](#_Toc40729454)

[2.1 Конволюционни невронни мрежи 8](#_Toc40729455)

[2.1.1 Дефиниция 8](#_Toc40729456)

[2.1.2 Градивни блокове 9](#_Toc40729457)

[2.2 Обучаващ алгоритъм за невронни мрежи - Backpropagation 14](#_Toc40729458)

[2.3 Машини на поддържащите вектори (SVM) 16](#_Toc40729459)

[2.4 Метрики за валидация 19](#_Toc40729460)

[2.4.1 Точност на класификация (accuracy) 19](#_Toc40729461)

[2.4.2 Матрица на грешките (confusion matrix) 19](#_Toc40729462)

[2.4.3 F1 резултат 20](#_Toc40729463)

[2.4.4 Средна абсолютна грешка 21](#_Toc40729464)

[2.4.5 Средна квадратична грешка 21](#_Toc40729465)

[3 Използвано множество от данни (Dataset) 22](#_Toc40729466)

[4 Архитектура на системата 25](#_Toc40729467)

[5 Резултати от работата на системата 26](#_Toc40729468)

[6 Възможности за бъдещо развитие 27](#_Toc40729469)

[7 Използвана литература 28](#_Toc40729470)

# Увод

Автоматичното засичане на лицеви изображение се превръща във все по важна зона за научни изследвания. Задачата включва в себе си подзадачи свързани със компютърното виждане, машинно саомообучение и поведенчески науки, и намира много приложения в областта на сигурността, взаимоотношението човек-компютър, сигурност на шофьори, и здраве опазване. През изминалите години има направен голям напредък в областта, с интерес към разпознаването на лица които не са позирали, и са поставени във реалистични среди, или позиранне от много различни ъгли.

В същността си задачата е класификационен проблем със входен параметър снимка, и изход един от няколко етикета, които показват емоцията на подадената снимка. Емоциите над които се фокусираме в тази дипломна работа са 7 – ярост, отвръщение, страх, щастие, тъга, изненада и презрение. Тези Емоции се засичат на базата освен на входна снимка и на подадени опорни точки (68 на брой, всяка със х, у координати) и информация за FACS актиавции, на база на опорните точки.

## Дефиниция за „Разпознаване на емоции“

Разпознаване на емоции е процесъс по идентифициране на човешката емоция. Хората често варират в тяхната точност по разпознаване на емоциите на други хора. Разпознаването на емоция използвайки технологии е сравнително нова област на науката. За момента, този проблем се решава най-добре ако човешкото лице бива заснемано от няколко позиции, използвайки видео и анотации от на лицата от професионалисти.

Точността на разпознаването на емоции се увеличава когато комбинира различни източници на човешките изражение, като текст, аудио или видео. Различни видове емоции могат да бъдат засечени чрез интеграцията на информация от лицеви изражения, движения на тялото и жестове и реч. Съществуващите подходи във разпознаването на емоции могат да бъдат разделени образно казано на три основни категории: техники базирани на знания, статистически методи или хибридни подходи.

**Техниките базирани на знания** използват знания за семантичните и синтактични характеристики, с цел да определят конкретни видове емоция. В този подход често се използват ресурси базирани на знания по време на класификацията на емоцията от типа на WordNet, SenticNet, ConceptNet, EmotiNet и други. Едно от предимствата на този подход е достъпността която идва със широката наличност на такива ресурси (базирани на знания). От друга страна недостатък на този подход е невъзможността му да се справя със концептуални нюанси и сложни лигвистични правила.

Подходите базирани на знания могат да бъдат класифицирани в 2 категории: базирани на речници и базирани на корпус от документи. Тези които са базирани на речници намират думи които показват мнение или емоция в речник и търсят за техни синоними и антоними за разпиряване на началният списък от възможни емоции. Корпус базираните подходи започват със стартов списък от думи които изразяват мнение или емоци, и разширяват базата намирайки други думи със контекстно специфични характеристики в по-голям корпус. Докато подходите базирани на корпос от документи взимат предвид контекста, те все още имах проблеми със точността в различните домейни тъй като дума в един домейм може да има различна ориентация и значение от това в друг домейн.

**Статистически методи** често включват използването на различни алгоритми за машинно самообучение, при които голям набор от анотирани данни се подава на алгоритъма и по този начин системата се научава да предстазва правилните видове емоция. Алгоритмите базирани на машинно самообучение в повечето случаи предоставят по-точни класификационни резултати, сравнено с другите подходи, но едно от предизвикателствата на тези подходи е намирането на достатъчно голяма обучаваща извадка.

Някои от най-често иползваните алгоритми за решаване на този проблем включват машини на поддържащите вектор, найвен Бейсов класификатор и максимум ентропия. Дъплбокото самообучение също се използва широко в разпознаването на емоции. Добре известни архитектури които решават такива типове проблеми са конволюционните нервонни мрежи, машини за екстремно обучение и дълго кратко трайна памет (long short-term memory LSTM). Популярността на подходите за дълбоко самообучение в тази област се дължи от част и на успеваемостта на тези подходи във полетата на компютърн зрение, разпознаване на реч и обработка на естествени езици.

**Хибридни подходи** при разпознаването на емоции са комбинация от техники базирани на знания и статистически подходи, които експроатират допълващи се характеристики за твата вида техники.

## FACS Дефиниция

FACS съкратено от facial action coding system или система за закодиране на лицевите движения, е система която има за цел да класифицира лицевите движения, по начинът им на деформация на лицето. Системата е базирана на работата на шведски анатомист на име Карл-Херман Хйортсьо. По-късно е била усвоена от Поул Екман и Уолъс В. Фриесен.

Движенията на всеки индивидуален мускул на лицето са закодирани с кодове, така че да хващат и най-малките промени в лицевото изражение. Често срещано е емоциите да бъдат закодирани със FACS кодове. Този подход намира приложения в работата на психолози и художници-аниматори.

Използвайки FACS, хората които определят кои кодове са активирани могат ръчно да опишат почети всяко лицево изражение което е анатомично възможно. Това става като то се деконструктира във спеицфични единици на действие (Action Units – AU). Тези единици са независими от от какъвто и да е вид интерпретация, и могат да бъдат използвани за вземане на решения за това каква емоция е налична. Наръчникът за FACS кодове е 500 страници и съдържа единиците на действие (AU), както и интерпретацията на Екман за значението на всяка една от тях.

Въпреки че определянето на всяко от човешките изражения изисква ръчна обработка от обучени експерти, учените имат успех и със определяне на FACS кодове използвайки софтуер. CANDIDE или Artnatomy са примери за такива системи, който позволяват израженията на човешкото лице да бъдат изкуствено създадни, използвайки нужните action unit-и.

Използването на FACS е предложено и за анализи на депресия, или измерване на болка на пациенти, които не могат да изразят болка вербално.

FACS индексира лицевите изражения, но не предоставя никаква био-механична информация за степента на активация на деден мускул на лицето. Въпреки че мускулната активация не е част от FACS, основните мускули включени във лицевото изражения са добавени.

Оценка на интензитета на FACS се анотира добавяйки букви А до Е към Action Unit-а с цел - да се покаже колко „силно“ е активиран AU. Пример:

А – лека следа

В – умерено загатнато

С – маркирано

D – усилено

E – максимално

Използвайки тези букви за индикация на интензитета, Action Unit-ите изглеждат могат да изглеждат по следния начин: 1Е, 2B, 3E и т.н.

Списък от AUs и техните описания.

|  |  |  |
| --- | --- | --- |
| КОД | ИМЕ | АКТИВНИ МУСКУЛИ |
| 0 | Neutral face |  |
| 1 | Inner brow raiser | [frontalis](https://en.wikipedia.org/wiki/Frontalis_muscle) ([pars medialis](https://en.wikipedia.org/wiki/Pars_medialis)) |
| 2 | Outer brow raiser | [frontalis](https://en.wikipedia.org/wiki/Frontalis_muscle) ([pars lateralis](https://en.wikipedia.org/wiki/Pars_lateralis)) |
| 4 | Brow lowerer | [depressor glabellae](https://en.wikipedia.org/wiki/Depressor_glabellae), [depressor supercilii](https://en.wikipedia.org/wiki/Depressor_supercilii), [corrugator supercilii](https://en.wikipedia.org/wiki/Corrugator_supercilii) |
| 5 | Upper lid raiser | [levator palpebrae superioris](https://en.wikipedia.org/wiki/Levator_palpebrae_superioris), [superior tarsal muscle](https://en.wikipedia.org/wiki/Superior_tarsal_muscle) |
| 6 | Cheek raiser | [orbicularis oculi](https://en.wikipedia.org/wiki/Orbicularis_oculi) ([pars orbitalis](https://en.wikipedia.org/wiki/Orbital_part_of_frontal_bone)) |
| 7 | Lid tightener | [orbicularis oculi](https://en.wikipedia.org/wiki/Orbicularis_oculi) ([pars palpebralis](https://en.wikipedia.org/wiki/Pars_palpebralis)) |
| 8 | Lips toward each other | [orbicularis oris](https://en.wikipedia.org/wiki/Orbicularis_oris) |
| 9 | Nose wrinkler | [levator labii superioris alaeque nasi](https://en.wikipedia.org/wiki/Levator_labii_superioris_alaeque_nasi) |
| 10 | Upper lip raiser | [levator labii superioris](https://en.wikipedia.org/wiki/Levator_labii_superioris), [caput infraorbitalis](https://en.wikipedia.org/wiki/Levator_labii_superioris) |
| 11 | Nasolabial deepener | [zygomaticus minor](https://en.wikipedia.org/wiki/Zygomaticus_minor) |
| 12 | Lip corner puller | [zygomaticus major](https://en.wikipedia.org/wiki/Zygomaticus_major) |
| 13 | Sharp lip puller | [levator anguli oris](https://en.wikipedia.org/wiki/Levator_anguli_oris) (also known as [caninus](https://en.wikipedia.org/wiki/Caninus" \o "Caninus)) |
| 14 | Dimpler | [buccinator](https://en.wikipedia.org/wiki/Buccinator) |
| 15 | Lip corner depressor | [depressor anguli oris](https://en.wikipedia.org/wiki/Depressor_anguli_oris) (also known as [triangularis](https://en.wikipedia.org/wiki/Triangularis)) |
| 16 | Lower lip depressor | [depressor labii inferioris](https://en.wikipedia.org/wiki/Depressor_labii_inferioris) |
| 17 | Chin raiser | [mentalis](https://en.wikipedia.org/wiki/Mentalis) |
| 18 | Lip pucker | [incisivii labii superioris](https://en.wikipedia.org/wiki/Incisivii_labii_superioris) and [incisivii labii inferioris](https://en.wikipedia.org/wiki/Incisivii_labii_inferioris" \o "Incisivii labii inferioris) |
| 19 | Tongue show |  |
| 20 | Lip stretcher | [risorius](https://en.wikipedia.org/wiki/Risorius) w/ [platysma](https://en.wikipedia.org/wiki/Platysma) |
| 21 | Neck tightener | [platysma](https://en.wikipedia.org/wiki/Platysma) |
| 22 | Lip funneler | [orbicularis oris](https://en.wikipedia.org/wiki/Orbicularis_oris) |
| 23 | Lip tightener | [orbicularis oris](https://en.wikipedia.org/wiki/Orbicularis_oris) |
| 24 | Lip pressor | [orbicularis oris](https://en.wikipedia.org/wiki/Orbicularis_oris) |
| 25 | Lips part | [depressor labii inferioris](https://en.wikipedia.org/wiki/Depressor_labii_inferioris), or relaxation of [mentalis](https://en.wikipedia.org/wiki/Mentalis) or [orbicularis oris](https://en.wikipedia.org/wiki/Orbicularis_oris) |
| 26 | Jaw drop | [masseter](https://en.wikipedia.org/wiki/Masseter); relaxed [temporalis](https://en.wikipedia.org/wiki/Temporalis) and [internal pterygoid](https://en.wikipedia.org/wiki/Medial_pterygoid_muscle) |
| 27 | Mouth stretch | [pterygoids](https://en.wikipedia.org/wiki/Pterygoid_bone), [digastric](https://en.wikipedia.org/wiki/Digastric) |
| 28 | Lip suck | [orbicularis oris](https://en.wikipedia.org/wiki/Orbicularis_oris) |

Примери за закодиране на емоции:

|  |  |
| --- | --- |
| Емоция | Списък от AUs |
| Гняв | 4+5+7+23 |
| Презрение | R12A+R14A |
| Отвръщение | 9+15+16 |
| Страх | 1+2+4+5+7+20+26 |
| Щастие | 6+12 |
| Тъга | 1+4+15 |
| Изненада | 1+2+5B+26 |

Пример за разчитане на таблицата – за да бъде класифицирана дадена емоция като „Гняв“ е нужно да бъдат активирани следните 4 единици на действия – 4 5 7 23, тоест да бъде занижена вътрешната част на веждата (4), да бъде повдигнати горните клепач (5), клепачите да бъдат стегнати (7), да бъдат стегнати устните (23).

## Цел на дипломната работа

Целта на настоящата дипломна работа е изграждане на система, която да може да класифицира емоцията на подадена снимка. Ще бъдат разгледани два подхода за имплементация, и ще бъде направено сравнение на резултатите между тях.

# Преглед на използваните научни методи

## Конволюционни невронни мрежи

Невронните мрежи са множество от алгоритми, които са моделирани по подобие на човешкият мозък, и се използват за разпознаване на шаблони. Използват се във класификационни и регресивни проблеми, шаблоните които те разпознават са цифрови, съдържащи се във вектори, във които трябва да се преведе всякакъв вид данни, от реалния свят – снимки/звук/текст.

### Дефиниция

В дълбокото самообучение, конволюционна невронна мрежа (CNN) е клас дъпбока невронна мрежа, най често използвана във разпознаването на визуални изображения. Те са известни със свойствата си да игнорират изместването и големината на детайлите които търсят. Това се базира на тяхната архитектура която е базирана на споделени тегла, и пренебрегването на транслациите на различни обекти. Имат приложения във видео и синкова обработка, препоръчващи системи, класификация на снимки, анализ на медицински снимки, обработка на естествени езици, обработка на финансови времеви серии и други.

Конволщционните невронни мрежи са регуларизирани версии на многослойните персептрони. Многослоен персептрон обикновенно означава напълно свързана мрежа, т.е. всеки неврон в един слой е свъран с всички неврони в следващият слой. Тази пълна свързаност на мрежите ги прави лесни за пренагаждане към данните. Типични начини за регуларизация включват някаква форма на загуба на теглата, или деактивация на някой от невроните. CNN има различен подход към регуларизацията, те се възползват от йерархичните шаблони в данните и съставят по-сложни шаблони, използвайки по-малки под-шаблони. Затва по отношение на размерност и свързаност, CNN едно ниво по-ниско от персептроните.

Вдъхновението за CNNs идва от биологичните процеси, при които шаблонат на свързаност между невроните наподобява организацията на животинската зрителна кора. Индивидуалните неврони от кората отговарят на стимулации само във ограничен регион от визуалното поле, известно като рецептивно поле. Тези рецептивни полета на различни неврони частично се застъпват, така че покриват цялото зрително поле.

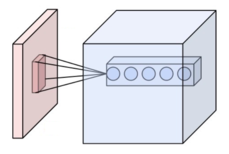
CNN нямат нужда от голяма преработка на данните, за разлика от други алгоритми за класификация на снимки. Това означава че мрежата сама научава филтри, които при традиционните алгоритми трябва да бъдат написани на ръка. Тази независимост от предишно знание, и ръчно напасване намаля нужната работа на програмиста, и позволява по-бърза разработка в дизайна на такъв тип мрежи.

### Градивни блокове

Архитектурата на CNN мрежите се формира чрез наслагване на различни слоеве които трансформират като му прилагат някаква диференцируема функция. Няколко различни типа слоеве са най-често използвани. По-долу са описани най-често използваните:

#### Конволюционен слой

Конволюционният слой е основният градивен блок на CNN мрежата. Параметрите на слоя се състоят във набор научаеми филтри (или Kernels-ядро), които имат малко поле на рецептивност, но обикалят по целият поток от входни данни. Повреме на предаването на информация напред, всеки филтър се прилага по цялата дължина и ширина на входните данни, и накрая изгражда двузимерна карта с активации на този филтър. Като резултат, мрежата научава филтри които се активират когато бъдат засечени някакви характерни форми във входните данни.



Фигура 2‑1Невроните на конволюционен слой(в синьо), свързани към съответстващоот им рецептивно поле(червено)

Когато работим със входни данни със големи измерения, каквито са снимките, не е практично да свързваме всички неврони към тези в предишният слой, защото такава мрежова архитектура не взема под предвид пространствената структура на данните. Конволюционните мрежи адресират този проблем като налагат локална свързаност която е ограничена между невроните от съседните слоеве – всеки неврон е свързан само със малък регион от входни данни.

Това нещо може да бъде контролирано чрез хиперпараметър за слоят, наречен рецептивно поле на неврона. Връзките са локални по пространство, но винаги обхващат цялата площ на входните данни.

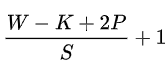


Фигура 2‑2Типична CNN архитектура

Три хиперпараметъра контролират размерът на изходните данни, които преминават през конволюционният слой – stride, depth и zero-padding.

* Stride (крачка) – кнтролира колко колони да бъдат алокирани във изходният формат. Когато този параметър е със стойност 1, тогава местим филтрите със 1 пиксел на всяка крачка. Това води до силно наслагване на рецептивните полета и също до изходни данни с голям обем. Когато параметърат е 2, тогава филтрите прескачат със 2 пиксела на крачка и т.н. Съответно – когато крачката е S > 0, филтърат прескача S на брой пиксела, когато обработва входните данни. На практика параметър с големина S > 3 се използва рядко.
* Depth (дълбочина) – Контролира броят на неврони във слоят които са свързани със същият регион във входнита данни. Тези неврони се учат да се активират за различни характерни черти във входните данни. Например, ако първият слой приема необработена снимка за вход, тогава различни неврони могат да се активират за да засекат наличието на конкретни ъгли или цветове.
* Zero-padding (отстояние) – Понякога е подходящо да се поставят нули във краят на входните данни, ако форматът налага някои пиксели да не бъдат обработени. Например ако имаме входни данни с формат 100х100px и размер на филтъра – 3px, то последният пиксел никога няма да бъде обработен. В такъв случаи можем да добавим padding = 1, кото би добавило по 1 пиксел в ляво и дясно (който ще е с нулева стойност) и ще можем да обработваме винаги и последният пиксел.

За изчисляване на изходният формат на конволюционният слой, използваме тези 3 параметъра, по следният начин W = големина на филтъра, S = крачка, P = големина на нулите които да сложим по рамката (zero-padding).



Формула 1 Изчисляване на изходния формат от конволюционен слой

#### Пулинг слой(Pooling layer)

Друга важна част от CNN е пулинга или слой за обединяване, което е форма на нелинейно намаляващо селектиране. Има няколко не линейни функции за имплементиране на пулинг, измежду които макс пулинга е най-популярен. То разделя входната снимка във множество от незастъпващи се правоъгълници, и за всеки под регион изкарва най-голямата стойност, като резултат.

Интуицията за този слой е следната – точната локация на даден входен параметър е по-малко важна, от колкото приблизителната му локация, в сравнение със други входни параметри. Това е идеята за използване на пулинг във конволюционните невронни мрежи. Пулинг слоят служи за прогресивното намаляне на размерноста на данните и по този начин да намали броят на параметрите, нужната памет и изчислителна сила за работата на модела. От тук следва и че той също ограничава да известна степен мрежата от пренагаждане. Честа практика е да се постави пулинг слой между последователни конволюционни слоеве във CNN архитектурата. Пулинг операцията също добавя още 1 форма на независимост към транслации. Пулинг слоят оперира независимо върху всяко парче от входните данните, и го мащабира съответно. Най-често срещаната форма на пулинг слой е 2х2 със крачка 2. Такъв тип операция премахва 75% от активациите на предният слой.



Формула 2: Пулинг операция

В този случай, всяка макс операция е върху 4 числа, и от тук идват тези 75% активации които се игнорират.

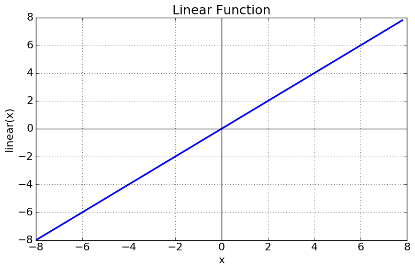
Алтернативи на макс пулинга, други фунцкии които могат да бъдат използвани са средно аритметичен пулинг или L2 нормиран пулинг. Средно аритметичният обикаля по всички входни числа, и взема средно аритметичното им, L2 прави евклидова нормализация на входните числа.

Тъй като използването на пулинг намаля драстично големината на входните данни, има тренд да се използват малки филтри, или понякога да се избягва ползването на пулинг слоеве.

#### Пълно свързан слой и видове активиращи функции

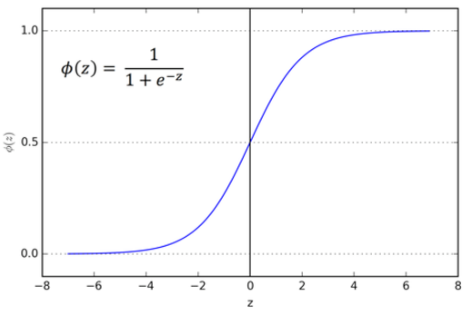
Накрая, след няколко конволюционни и пулинг слоя, логиката която научава как комбинацията от филтри разпознават даден образ се прави чрез пълно свързан слой. Невроните във пълно свързаният слой имат връзки към всеки неврон от предишния слой, точно както е във стандартните (не конволюционни) невронни мрежи. Във пълно свързаният слой имаме различни набори от активиращи функции, които можем да използваме. Ето и примери за някои активиращи функции:

* Линейна активация – f(x) = x с обхват (-∞;+∞). Това е най-простият вид активация. Може да бъде използван като изходна активация за последен слой, ако задачата която се решава е регресивен проблем.



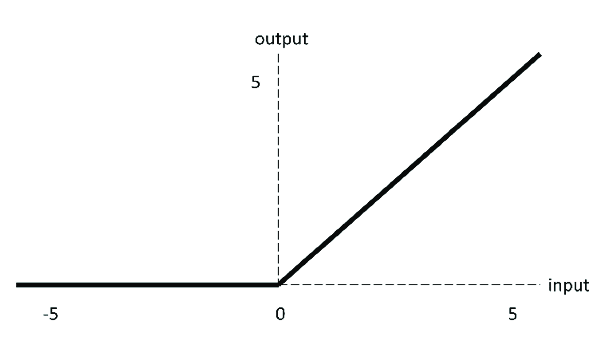
Фигура 2‑3 Линейна активация

* Сигмоидна активация – Основната причина поради която се използва този тип активация е защото обхватът и е от (0;1) за това много често тя се използва когато изходът трябва да бъде някаква вероятност. И тъй като вероятноста на каквото и да е е число между 0 и 1, сигмоидната активация е правилният избор за такъв тип проблем.



Фигура 2‑4 Сигмоидна активация

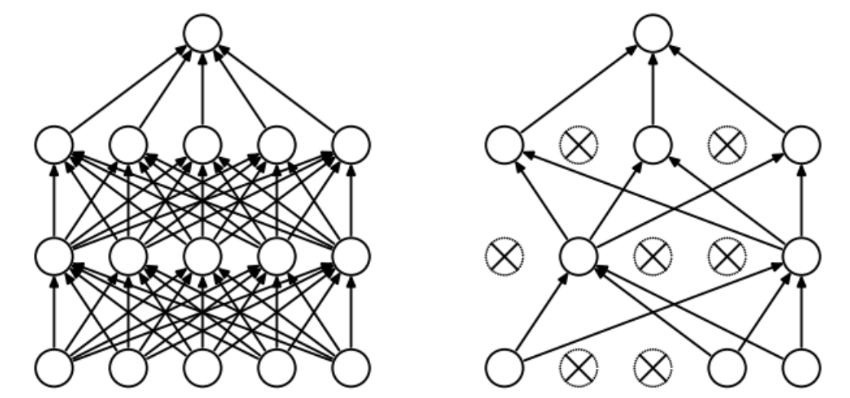
* ReLU активация – Това е най-често използваната активация за скрити слоеве в невронните мрежи. Формулата за ReLU е: f(x) = max(0, x) като предимствата и са няколко, първо има рядко разпределена активация при случайно инициализиране на невронната мрежа – около 50% от невроните се активират, второ по-добро предаване на грешката в задна посока по време на backpropagation – тази функция няма проблемът със изчезващият градиент както при сигмоидната активация например, трето – лесна за изчисление, нужните операции са само сравнение, събиране и умножение. Като недостатъци могат да се посочат следните – не е диференцируема в нулата, няма граница за максимална стойност, не е центрирана около нулата.



Фигура 2‑5 ReLU активация

#### Dropout слой

Честа практика е пълно свързаните слоеве да бъдат следвани от dropout слой. Този вид слой има за цел да адресира проблема със пренагаждането към обучаващите данни във невронните мрежи. Той работи като на всеки неврон от предишният слой (пълно свързаният) се добавя вероятност да бъде изпуснат в следващия стадии на изчисления. По този начин някои тегла и връзки се губят, и невроните се научават да работят самостоятелно един от друг, и да научават по-общи поведения. Пример за работата на dropout слой може наглендо да се види в следващата фигура.



Фигура 2‑6: Ляво - невронна мрежа без Dropout. Десно - със Dropout

## Обучаващ алгоритъм за невронни мрежи - Backpropagation

В машинното обучение, backpropagation или алгоритъм с обратно разпространение на грешката е най-широко използваният алгоритъм за обучение на невронни мрежи за обучение с учител. Backpropagation съществува като генерализиран тип алгоритми за обучение на невронни мрежи и като цяло за различни видове функции. По време на напасването на невронната мрежа към данните, backpropagation ичислява градиента на функцията която оптимизиране по отношение на теглата на мрежата, за всеки входно-изходен пример. Ефективността на този метод го прави подходящ за обучение на многослойни мрежи, използвайки градиентни методи и обновявайки теглата за минимизиране на целевата функция – градиентно спускане и стохастично градиентно спускане са най-често използваните оптимизационни похвати. Backpropagation работи изчислявайки градиента на целевата функция по отнишение на теглата следвайки верижното правило, изчислявайки градиента на всеки един слой, итерирайки назад от последният слой за избягване на излишни изчисления на междинните стойности във верижното правило.

Терминът backpropagation се отнася стрикно само за алгоритъма за изчисляван на градиента, а не за това как градиента се използва, но терминът често се използва за целият обучаващ алгоритъм, включително за това как градиента се използва. Backpropagation генерализира изчислението на градиента във делта правилото, което е еднослойната версия на backpropagation.

Преглед на параметрите и алгоритъма.

Нека Х – входен вектор от атрибути

У – изходен вектор ( за класификация изходът ще съдържа вероятности, например (0.1, 0.7, 0.2) а целевият вектор ще бъде “one-hot encoded” със стойности (0, 1, 0)

С – целева функция. При класификациони проблеми това в повечето случаи е крос ентропия, докато за регресия най-често се ползва квадратна грешка.

L – брой слоеве

Wl – Теглата между слоеве l-1 и l

fl – Активационни функции на слой l. За класификационни задачи последният слой обикновенно използва логистична функция за двоична класификация и софтмакс за много класова класификация, докато скритите слоеве използваме ReLU

По време на изчислениято на производните на алгоритъма се използват и други междинни стойности, тези стойности са описани по-долу. За целите на backpropagation-а, конкретно избраната целева функция и активационни функции нямат значение, стига техните производни да могат да бъдат изчислени по ефективен начин. Целият процес е комбинация от умножение на матрици и композиция на функции. Крайният резултат g(x) може да бъде формализиран по следният начин:



За обучаващото множество ще бъдат налични двойки от входно-изходни стойности {(xi, yi)}. За всяка от входно изходните двойки загубата за модела се изчислява като разликата между предсказаният изход g(xi) и целевият изход yi:

C(yi, g(xi))

По време на изчислението на модела, теглата са фиксирано, докато входните данни варират (и целевият изход е неизвестен), и мрежата приключва със изходния слой (той не включва целевата функция). По време на обучението, входно-изходните двойки са фиксирани, докато теглата варират, и изчисленията приключват със целевата функция.

Backpropagation изчислява градиента за фиксирани входно-изходни двойки (xi, yi), където теглата W могат да варират. Всеки индивидуален компонент на градиента  може да бъде изчислен по вреижното правило за диференциране, но изчилението по този начин за всяко отделно тегло е неефективно. Ефективното изчисление на градиентите избягвайки дублиращи се изчисления и изчисляванто на междинните стойности се прави, като се изчислява градиента на всеки слой – по-точно, градиента на претегленият вход за всеки слой – отбелязан със dl отзад напред.

Неформално начинът по който дадено тегло W влияе на целевата функция е чрез ефектът си на следващият слой, и това става линейно, dl е единствената информация която е нужна за изчисляване на градиентите на теглата на слой l, и след това можем да изчислим предишният слой dl-1 и да повторим рекурсивно. Това решава проблема със не ефективността по 2 начина. Първо, избягва се дубликирането защото когато изчислим градиента на слой l, нямаме нужда да изчисляваме наново всички производни на следващите слоеве l+1, l+2 всеки път. Второ, избягват се ненужни междинни изчисления защото на всеки етап директно се изчисляват градиентите на теглата по отношение на крайният изход (на целевата функция), отколкото ненужното изчисление на производни и на слоевете на скритите слоеве по отношение на промените в теглата .

По отношение на целевата функция, тя е такава функция която съпоставя стойности на една или повече променливи към някакво реално число, представяйки някакъв вид „цена“ асоциирана със тези стойности. За backpropagation, целевата функция изчислява разликата между изходът на мрежата и желаният изход, след като даден пример е преминал от край до край.

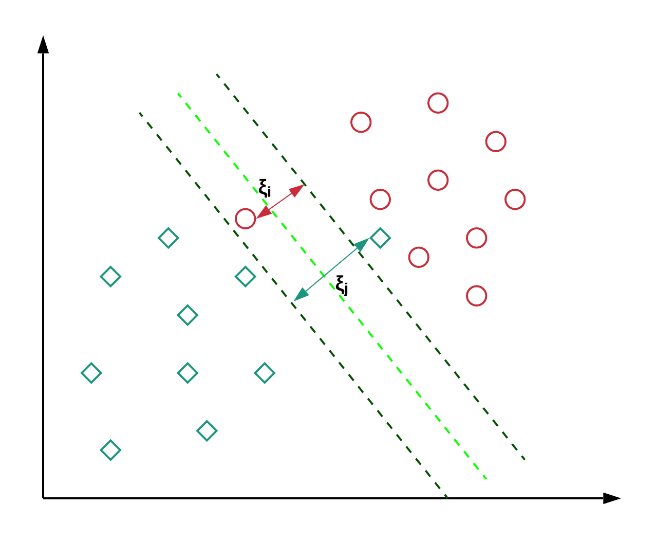
Математическите свойства които трябва да притежава целевата функция са 2, за да може да бъде използвана във backpropagation. Първо е че трябва да бъде записана като средното  върху дадени функции които изчисляват грепки Ex за n на брой обучаващи примера x. Нуждата от това свойство е че backpropagation изчислява градиента на целевата функция за един пример, който трябва да се генерализира към общата целева функция. Второто нужно свойство е тя да може да бъде записана като функция на изходите на невронната мрежа. Примери за целеви функции са крос ентропия, средна грешка и средна квадратична грешка.

Като ограничения на алгоритъма може да бъде посочено че няма гаранция че той винаги ще намери глобалният минимум на целевата функция, а само локален минимум. Също така съществуват проблеми със минаването през „плата“ на функцията – това ще рече продължителни проски участъци.

## Машини на поддържащите вектори (SVM)

В машинното самообучение, машина на поддържащите вектори(SVM) представлява метод за учение с учител който може да бъде използван за решаване на класификационни и регресивни проблеми. При даден набор от обучаващи примери, всеки от които маркиран като принадлежащ на една от 2 категории, SVM обучаващият алгоритъм изгражда модел, който присвоява на новите примери една от двете начално зададени категории, правейки го не вероятностен двуичен линеен класификатор.

SVM моделът представя примерите като точки в n-мерното пространство, така разпределени, че примерите от отделни категории са разделени от права, и отстоящото разстояние на примерите до правата е възможно най-голямо. Новите примери след това се поставят в същото пространство, и предсказването се определя в зависимост от коя страна на научената граница стои новият пример.



Фигура 2‑7: SVM разделяне на множества

В допълнение към извършването на линейна класификация, SVM може също да извършва не линейна класификация, използвайки различни видове ядра (kernels).

По-формалнo, машините на поддържащи вектори конструират хиперравнина, или множество хипер равнини във многомерното пространство, които се използват за класификация или регресия. Добро разделяне на данните се постига когато хиперравнината която има най-голямо отстояние от най-близките обучаващи точки, на който и да е от класовете. От тук следва и че колкого по-голямо е отстоянието, толкова по-малка е и грешката от генерализация на класификатора.

При подадено обучаващо множество от n на брой точки,във формата (x1, ­y1),…,(xn, yn) където yi е или 1 или -1, всеки индикиращ класът към който точката xi принадлежи. Всеки xi е p-измерен вектор от реални числа. Искаме да намерим хипер равнина с максимално отстояние, която разделя групата точки xi за които yi = 1 от групата от точки за които yi = -1. Всяка хиперравнина може да бъде изразена като множество от точки x удовлетворяващо следното уравнение:



Където w е нормалният(нормализиран) вектор към хиперравнината. Това е като нормалната форма на Хесе, с изключение че W не е задължително да бъде базисен вектор. Параметърът b/||w|| определя отстоянието на хиперравнината от центъра по отношение на нормализирания вектор w.

Ако обучаващите данни са линейно разделими, можем да изберем 2 паралелни хиерравнини които разделят двата класа данни, така че разстоянието между тях да е възможно най-голямо. Регионът който е между тези 2 хиперравнини наричаме „отстояние“, и хиперравнината която е със максимално отстоние от двата класас е тази която стои посредата на тези 2 равнини. Със нормализирано или стандартизирано множество данни, тези хиперравнини могат да бъдат описани със следните уравнения

 всичко на или над тази равнина е от клас 1

 всичко на или под тази равнина е от клас -1

Геометрично, разстоянието между тези 2 хиперравнини е 2/||w|| така че за да максимизиране разстоянието между равнините искаме да минимизираме вектора w. Разстоянието се изчислява използвайки уравнението за отстояние на точка до равнина. Също така ограничаваме данните от попадането върху отстоянието, като дабавяме следните ограничения за всяко i:





Тези ограничения могат да бъдат записани като



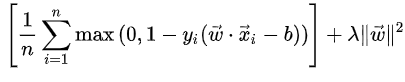
Тези ограничения налагат всяка точка да лежи на правилната страна на хиперравнината. С това оптимизационният ни проблем се првръща в минимизиране на векторът w по отношение на .

Стойностите на w и b които решават този проблем представляват нашият класификатор. Важна последица от това геометрично описание е че хиперравнината с максимално разстояние е изцяло определена от тези точки x които лежат нак-близо до нея. Тези точки x се наричат поддържащи вектори.

За разширяване на случаите в които данните не са линейно разделими се използва hinge loss функция.



Тази функция е 0 ако ограничението в (1) е изпълнено, с други думи ако x лежи на правилната страна на хиперравнината. За данни от грешната страна на равнината, стойноста на функцията е пропорционална на отстоянието от хиперравнината. От тук ние се стремим да оптимизираме:



Където параметърът λ определя нарастващото отстояние и осигурява че x е от правилната страна на хиперравнината. За това за значително малки стойности на λ вторият параметър на целевата функция ще стане пренебрежимо малък, и от тук ще се държи както във случая със линейно разделимите данни.

## Метрики за валидация

### Точност на класификация (accuracy)

Точността на класификация обикновено е това, което се има предвид, когато се използва терминът точност. То е съотношението между правилно предсказаните стойности и броя на всички направени прогнози.



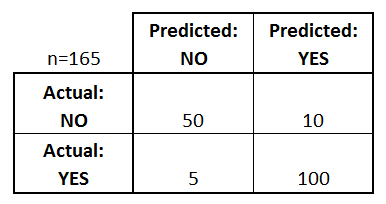
Работи добре единствено ако има равен брой обучаващи примери, принадлежащи към всеки клас. Например, в случай, че в обучаващото множество има 98% обучаващи примери от клас А и 2% от клас В, нашият модел лесно би могъл да постигне 98% тренировъчна точност просто като прогнозира, че всеки пример принадлежи към клас А.

Когато същият модел се тества с обучавао множество, в което 60% от примерите са от клас А и 40% от клас В, то тогава тренировъчната точност би спаднала до 60%. Това показва, че точността на класификация би могла да даде фалшива увереност в постигнатата точност.

Истинският проблем възниква, когато цената на грешното класифициране на обучаващите проби от непреобладаващия клас е висока. Например, ако се обработват данни за рядко, но смъртоносно заболяване, цената на грешното диагностициране на болен човек като здрав е много по-висока, отколкото ако се наложи здрави хора да бъдат подложени на допълнителни тестове.

### Матрица на грешките (confusion matrix)

Матрицата на грешките както името подсказва ни дава матрица като изход, и описва пълното поведение на модела ни. Ако предположим че имаме двуичен класификационен проблем имаме примери принадлежащи към 2 класа – yes или no. Също така имаме и наш класификатор който предсказва клас за подаден пример. При тестването на нашият модел на 165 примера получаваме следният резултат:



Фигура 2‑8 Пример за матрица на грешките

Тук виждаме 4 важни неша:

**Истински позитивни(TP)** – това са случаите в които ние сме предсказали YES и данните също са имали стойност YES (100 примера)

**Истински негативни(TN)** – Това са случаите в които сме предсказали NO и данните също са имали стойност NO (50 примера)

**Грешно позитивни(FP)** – това са случаите в които ние сме предсказали YES а всъщност е трябвало да се предскаже NO (10 примера)

**Грешно негативни(FN)** – това са случаите в които ние сме предсказали NO а всъщност е трябвало да се предскаже YES (5 примера)

Точността на матрицата ни се изчислява като се вземе стредно аритметичното на всички стойности лежащи на главният диагонал, в този случай = (TP + TN) / Всички примеи. Този тип матрица е базата за оценка на другите видове метрики.

### F1 резултат

F1 резултатът е хармоничното средно между precission и recall. Стойностите на F1 са в интервала [0, 1]. То казва колко прецизен е класификаторът (колко входни данни класифицира коректно) и също така колко устойчив е (не пропуска оглям брой примери).

Висок precision но нисък recall, дава голяма точност, но след това пропуска голям номер от примери които са трудни за класифициране. Колкото по-голям е F1 резултатът, толкова по-добре се справя моделът ни. Тази метрика може да ес изрази по следния начин:



**Precision(Прецизност) –** Наричаме броят на кореткните поситивни резултати, разделени на броят на позитивните резултати предсказани от класификатора.



**Recall(Отзование)** – Наричаме броят н коретните положителни резултати, разделен на номера на всички релевантни примери (всички които е трябвало да бъдат идентифицирани като позитивни).



### Средна абсолютна грешка

Средната абсолютна грешка е средното между разликата в оригиналните стойности и предсказаните стойности. Този вид метрика се използва при регресивни проблеми. То ни дава мярка за това колко на дълече са отишли предсказанте стойности от реално желаните. Този тип грешка обаче не ни дава идея за посоката на грешката, т.е. дали сме предсказали по-ниска стойност или по-висока стойност. Математически се представя така:



### Средна квадратична грешка

Този тип грешка е подобна на средно абсолютната грешка, разликата е че квадратичната грешка взема средното на квадратите на разликата между оригиналните и предсказаните стойности. Предимството е че можем по-лесно да изчислим градиента, докато при средната абсолютна грешка са ни нужни някой инструменти от линейното програмиране за да се справим с проблема. Тъй като вземаме квадратът на грешката, ефекта на голямата грешка става по голям, от този на малката грешка, и от тук моделът се фокусира повече върху поправянето на големите грешки.



# Използвано множество от данни (Dataset)

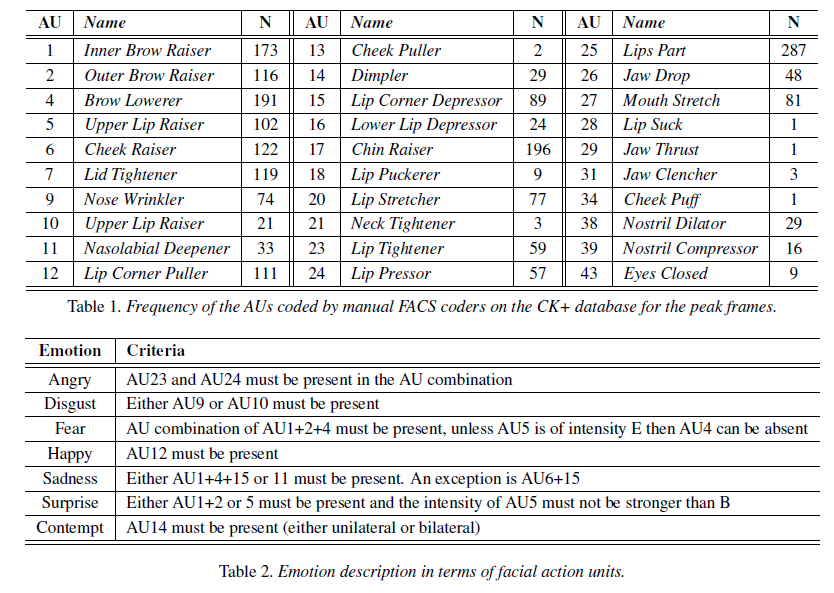
През 2000 година Cohn-Kanade dataset-a е съставен с цел промотирането на изследвания в областта на автоматичното разпознаване на лицеви изображения. От токава CK базата става една от най-широко използваните бази за сравнение за алгоритми в тази сфера.

Автоматичното засичане на лицеви изражения се превръща в сфера с нарастващо значение. Проблемът включва компютърно зрение, машинно само обучение и поведенчески науки, и може да бъде използвано на много места, като например сигурност, интеракция човек-компютър, сигурност на шофьорите, и здраве опазване.

В използваната диструбуция CK базата съдържа 593 сесии на 123 човека. Всяка сесия съдържа снимки, като стартовата снимка е във неутрално изражение, а последната снимка – в пиково изражение. Пиковият кадър е анотиран със FACS (facial action coding system).

Лицевото поведения на 210 възрастни бива записано използвайки синхронизирани камери. Участниците са на възраст от 18 до 50 години, 69% жени, 81% Евро-Американци, 13% Афро-Американци, 6% други етнически групи. Участниците са инструктирани от човекът провеждащ експеримента да извършат серия от 23 лицеви изображения. Тези включват единични FACS маркери, или комбинация от FACS маркери. Снимките са във формат 640х480рх със 8bit черно-бяла скала.

За всичките 593 сесии, има предоставено FACS кодиране, на пиковият кадър. Приблизително 15% от тези сесии са кодирани от втори сертифициран FACS кодер.



В горе посочената таблица се виждат кодовете на Action Unit-ите, например 1 – вътрешната вежда е повдигната, 2 – външната вежда е повдигната, видовете емоциите (7 на брой) и нужният критерий за да бъде категоризирана дадане емоция във снимка като активна.



Фигура 3‑1 Примерни изражения от Cohn-Kanade dataset-a

Освен FACS кодирането, всяка снимка е анотирана със 68 на брой опорни точки (landmarks) всяка от които със X и Y координати. За пример – в долната снимка, с червени кръгчета са отбелязани 68те опорни точки.



Фигура 3‑2 Примери за опорни точки на дадено изражение

# Архитектура на системата

# Резултати от работата на системата

# Възможности за бъдещо развитие

# Използвана литература

1. The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit

[Фигура **Error! No text of specified style in document.**‑1: SVM разделяне на множества 6](#_Toc39692718)