Автоматика – рубриката горе

Въведение в конволюционните невронни мрежи

А. Христов, М. Нишева, Д. Димов

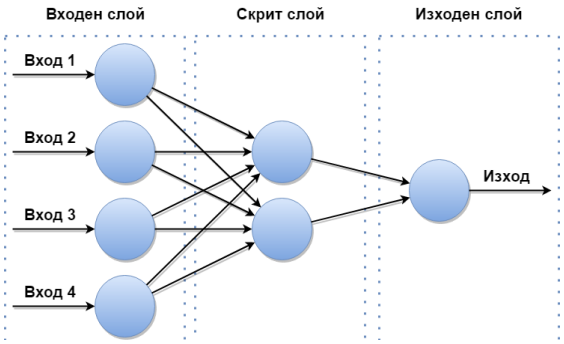
An Introduction into Convolutional Neural Networks

A. Hristov, M. Nisheva, D. Dimov

1. Увод

Изкуствените невронни мрежи (ANNs) [1] са изчислителни системи, вдъхновени от начина, по който функционират биологичните нервни системи (като човешкия мозък). ANN са съставени от голям брой взаимосвързани изчислителни възли (наричани по-нататък „неврони“), организирани в слоеве. Входните данни се предоставят на мрежата чрез входния слой, който комуникира с нула, един или повече скрити слоеве, където се извършва основната обработка на данните. Скритите слоеве след това се свързват с изходен слой, който извършва заключителна обработка и извежда резултата от изчисленията. В общия случай са допустими произволни обратни връзки между слоевете. Тук ще разглеждаме предимно *прави* (feed-forward) ANNs, един класически пример за които е даден на Фиг.1.

Проектирането на изкуствена невронна мрежа, предназначена за решаване на дадена конкретна задача, обикновено се извършва по следната обща схема. Най-напред се определя топологията на мрежата, т.е. броят на слоевете и броят на възлите във всеки слой. Броят на възлите от входния слой се определя от размерността на входните данни, а броят на възлите от изходния слой – от размерността на резултата. Броят и размерите на скритите слоеве се определят итеративно в зависимост от предметната област и конкретната задача. След определянето на топологията на мрежата се преминава към нейното *обучение*, т.е. към определянето на подходящи стойности на теглата на връзките между невроните.



Фиг. 1. Трислойна права невронна мрежа, съставена от входен, скрит и изходен слой от неврони.

Съществуват два основни типа обучение на ANNs:

- Обучение с учител, при което обучението се осъществява чрез предварително обозначени (анотирани) входни данни, които представят желания резултат. Всеки пример за обучение съдържа в себе си входен вектор от стойности и една или повече обозначени изходни стойности. Целта е да се минимизира цялостната класификационна грешка на модела чрез правилно изчисляване на изходната стойност.
- Обучение без учител (или самообучение), което се отличава с това, че входните данни за обучение не са предварително анотирани (обозначени). Успехът на обучението обикновено се определя от това дали мрежата е успяла да намали съответната функция на грешката (cost function), или респективно да увеличи съответната функция на печалбата (gain function).

Конволюционните невронни мрежи (CNNs) са вдъхновени от биологичните процеси, тъй като връзката между невроните наподобява на организацията на зрителната кора на бозайниците. CNNs са проектирани да откриват сложни характеристика във визуалните данни и напоследък бележат значителен напредък, главно в разпознаването на изображения. CNNs се обучават главно по методи с учител, т.е. самообучението, което е доста по-сложен и продължителен процес, е слабо застъпено при CNN, поне засега.

От изследванията на Hubel и Wiesel при маймуни (и котки) [2] е известно, че мозъкът съдържа малки региони от клетки, чувствителни към определени региони от зрителното поле, наречени поле на възприятие (receptive field). Те са позиционирани така, че да покриват цялото зрително поле. Тези клетки действат като филтри над ограничено входно пространство, откриващи и извличащи единствено локални връзки от съответната област.

Идентифицирани са два основни вида клетки:

- ♦ Обикновените клетки реагират максимално на специфични ръбовоидни модели в рамките на своето поле на възприятие.
- ♦ Комплексните клетки имат по-широки полета на възприемчивост и са локално инвариантни спрямо точната позиция на модела.

Тъй като зрителната кора на бозайниците е най-мощната съществуваща система за визуална обработка, изглежда естествено нейния модел на действие да бъде приложен в сферата на разпознаване на образи.

Името „конволюционни невронни мрежи“ подсказва, че мрежата съществено използва конволюционни слоеве. Тези слоеве са съставени от определен брой филтри, които равномерно обхождат изображението и проверяват за конкретна негова характеристика. Когато даден филтър открие специфичната характеристика, той задейства

съответния неврон от следващия слой. Обикновено, всеки неврон е свързан с малък регион от входни неврони, т.е. CNNs използват пространствено локална корелация чрез прилагане на локален модел (или модели) на свързване между невроните от съседни слоеве.

Целта на настоящата работа е да въведе читателя в CNNs чрез опростени техни архитектури и с акценти върху значимите разработки в областта, представяйки еволюцията им през годините.

Структура на обзора: В секция 2 са представени основните операции, използвани в CNN, чрез една типична CNN-архитектура и метода за обучение. Секция 3 съдържа резюме на най-значимите модели CNNs, а секция 4 дава сравнителен анализ. В заключителната секция 5 са дискусията и изводите от представеното разглеждане на CNN, от гледната точка на разпознаването на образи.

2. Структура и обучение на конволюционните невронни мрежи

Конволюционните невронни мрежи (CNNs) трансформират, слой по слой, пикселните стойности на изображенията в резултат, който ги класифицира към един от класовете, дефинирани заедно с тренировъчните входни данни. Някои слоеве извършват фиксирани математически операции; други съдържат параметри, настройвани (обучавани, тренирани) така, че резултатите изчислявани от мрежата, да съответстват (средно-статистически максимално) на класа, към който принадлежи съответното изображение на входа.

2.1. Архитектура на CNN

Слоевете от нерони, изграждащи една CNN, най-общо реализират 4 основни операции: конволюция, нелинейност (ReLU), обединяване (Pooling) и пълно свързване (Fully Connected), илюстрирани на Фиг.2:

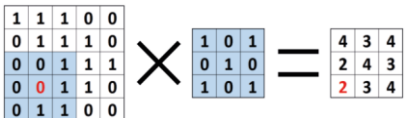
Фиг. 2. Примерна архитектура на CNN.

2.1.1. Конволюционен слой

Главната цел на конволюционния слой е да се извлекат различни характеристики (features) от входното изображение. В началните слоеве на мрежата, се извличат характеристики от ниско ниво, като ръбове, линии и ъгли. Крайните конволюционни слоеве пък извличат характеристики от високо ниво.

Всеки конволюционен слой е съставен от определен брой филтри (filters), които обхождат входните за него данни и проверяват за конкретна характеристика. На всяка позиция от обхождането се извършва скаларно умножение на 2 матрици (конкретния филтър и частта от входните данни, препокрити от него), а резултатът, т.е. сумата от поелементните умножения (евентуално нормирана) е изходът от филтрирането в дадената позиция. Резултатите от всички позиции, през които минава даденият филтър, образуват т.нар. *карта на характеристиките* (feature map). Така, броят на филтрите в даден конволюционен слой съответства на броя от карти на характеристиките, които се изчисляват в слоя.

На Фиг.3 е показано изчисляването на конволюция на изображение с размер 5×5 с филтър с размер 3×3. Тук филтърът обхожда (плъзга се върху) входното изображение със стъпка/отместване (stride) от по 1 пиксел, а резултатът, картата на характеристиките, е с размер 3×3. Очевидно, на всяка своя позиция филтърът „вижда“ само част от входното изображение.



Фиг. 3. Операция конволюция: Изходната матрица се нарича карта на характеристиките (feature map).

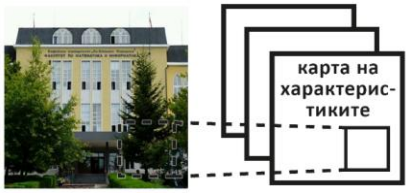
Фиг. 4. Пример за прилагане на (диференциален) филтър върху входно изображение.

Очевидно, различни филтри ще произведат различни карти на характеристиките от едно и също входно изображение. Пример за приложена конволюция може да се види във Фиг.4.

CNN „автоматично“ изчисляват съответните карти на характеристиките от слой към слой, но най-често някои от параметрите им се фиксират, например, видът, размерът и броят на филтрите, както и архитектурата на мрежата преди съответния слой. Колкото повече филтри се използват, толкова повече характеристични карти се извличат, което означава по-детайлно обучение на мрежата и предполага по-точно разпознаване (в „експлоатационен“ режим) след това.

Пълният обем на картите с характеристики се контролира от три параметъра, които трябва да се уточнят преди реализирането на конволюцията в слоя, а именно – дълбочина, отместване (стъпка на обхождане) и разширяване с нули:

- *Дълбочината d* съответства на броя на филтрите, използвани в конволюционния слой. На Фиг.5 е показана конволюция с 3 различни филтъра, от което се получават 3 различни карти на характеристиките. Тези 3 карти, разглеждани като свързани двумерни матрици, определят „пълната“ карта на характеристиките, с дълбочина d=3, в дадения слой. Въпреки, че не е задължително, размерът на филтрите в слоя обикновено се избира еднакъв, $n \times m$, при което пълната карта в слоя се явява 3-размерна матрица, $n \times m \times d$.



Фиг. 5. Операция конволюция: Карта на характеристиките с дълбочина 3 (тъй като са използвани 3 различни филтъра).

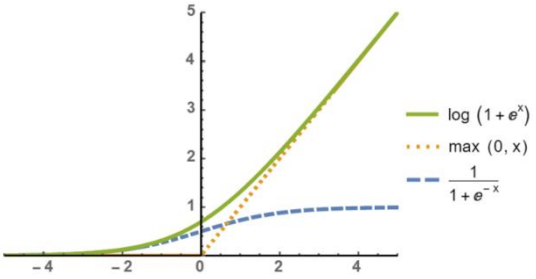
- *Отместването* е стъпката (в брой на пиксели), с която избраният филтър обхожда входната матрица. По-голямо отместване води до по-малки карти на характеристиките (и обратно).
- *Разширяване с нула* (zero-padding): Понякога е удобно входната матрица да се разшири с нули по периферията ѝ, за да може филтърът да обходи и границите ѝ.

Прочее, понятието „дълбочина“ се използва и спрямо цялата CNN, най-общо, отразявайки броя на слоевете ѝ.

2.1.2. Слой нелинейност (ReLU)

Проблемът с изчезващия градиент [3], представен от Sepp (Josef) Hochreiter показва, че при невронни мрежи с активационна функция като сигмоид или хиперболична тангента, където диапазонът на градиента е (-1, 1) или [0, 1), при мрежа с *N* на брой слоеве, градиентът намалява експоненциално с *N*. Това води до много по-бавно обучение на началните слоеве, спрямо останалите.

Активационните функции от типа „*ректифицирани линейни единици*“ (Rectified Linear Unit, ReLU) се появяват около 2000г., като възможно решение на проблема с изчезващия градиент. С изследванията на т.нар. „*зашумени*“ (noisy) ReLU, [4] от 2010г., се утвърждават 2 вида ReLU в областта, softPlus: $\sim \log(1+e^x)$, и нейната (често използвана) апроксимация softMax: $\sim \max(0, x)$. Ако отново погледнем Фиг.2, там ReLU са представени след всеки конволюционен слой. Фиг.6 предлага графично сравнение между софтплюс, софтмакс и конвенционалния сигмоид.

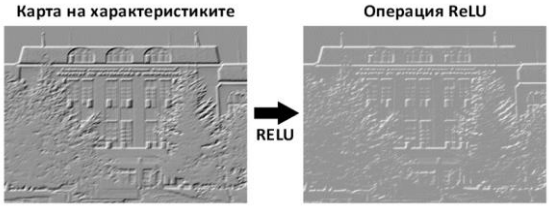


Фиг. 6. Съпоставяне между сигмоид и ReLU (софтмакс и софтплюс).

Главната разлика между сигмоида, характерен за класическите ANNs и ReLU е, че:

- Сигмоидът има ограничен диапазон [0, 1], докато „ефективният“ ReLU диапазон е [0, ∞).
- Градиентът на сигмоида изчезва (клони към нула) при големи стойности на входа *x*, докато при ReLU градиентът клони към константа, =1, в положителната посока за *x*.

ReLU се прилага върху всеки пиксел в дадения слой, като в резултат, всички негативни стойности ($\cong 0$), се заменят с нули, илюстрирано на Фиг.7. Преминала през ReLU, дадената карта на характеристиките се нарича вече „*ректифицирана*“.

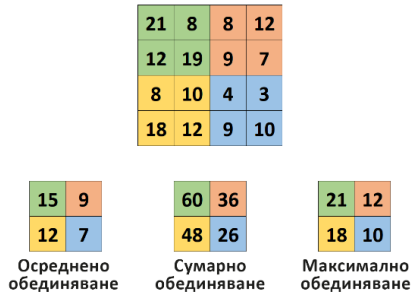


Фиг. 7. Операция ReLU (преди нея черните пиксели имат отрицателни стойности, а белите положителни).

Тъй като конволюцията е линейна операция, чрез ReLU се добавя възможността мрежата да моделира нелинейни функции. Могат да се използват и други, като хиперболична тангента или сигмоид, но се оказва, че ReLU се справя по-добре от тях в повечето ситуации.

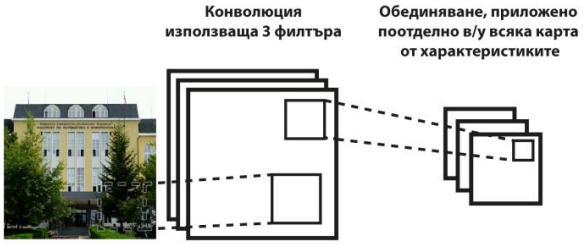
2.1.3. Слой обединяване (pooling)

Обединяването, приложено върху дадена карта на характеристики, съкращава размерността ѝ, но същевременно запазва (по-)съществената информация. Така характеристиките стават устойчиви срещу шум и изкривяване във входните данни. Обединяването може да бъде от различен тип: максимално (max pooling), осреднено (average pooling), сумарно (sum pooling) и т.н. Операцията е илюстрирана на Фиг.8, където при обединяване 2×2, входната матрица 4×4 се разделя на 4 неприпокриващи се региона с размери 2×2. В случай на максимално обединяване, резултатът е матрица на максималните стойности от всеки входен регион. При сумарното – резултатът са сумите от всеки входен регион, докато при осреднената – резултатът допълнително се нормализира (целочислено) с размера на входните региони.



Фиг. 8. Пример за осреднено, сумарно и максимално обединяване.

В мрежата, показана на Фиг.9, обединяващата операция е приложена поотделно върху всяка карта на характеристиките. Резултатът е 3 (компресирани) карти на характеристиките, изчислени от 3-те входни такива, т.е. имаме редукция на обема при запазване на дълбочината.



Фиг. 9. Илюстрация на операцията обединяване в CNN.

Идеята на обединяването е да се съкрати размера (обема) на входящите данни. По-важните свойства на операцията „обединяване“ са следните:

- Намалява размера на входните данни (и/или характеристики), откъдето следва и по-леката им управляемост (разбираемост);
- Намалява броя на параметрите и следователно, на изчисленията в мрежата;
- Прави съответния слой (и мрежата като цяло) инвариантна към малки трансформации на входа, като изкривявания, отмествания и др.;
- Представя входните изображения почти инвариантно спрямо мащаба (scale) и позицията на обектите в тях, което е много полезно на практика.

2.1.4. Напълно свързан слой

Това е традиционна ANN със софтмакс активационна функция в изходния слой. Терминът „напълно свързан“ показва, че всеки неврон от даден слой е свързан с всеки неврон от предишния слой. Изходът от конволюционните и обединяващите слоеве представя характеристики от високо ниво на входното изображение. Целта на напълно свързания слой е да използва тези характеристики, за да класифицира входното изображение към един от класовете определени от тренировъчните данни.

2.2. Обучение на CNN

Процесът на обучение на CNN чрез класическия Backpropagation метод може да бъде обобщен в следните 5 стъпки:

1: Инициализиране на всички филтри и параметри-(тегла) със случайни стойности.
2: Мрежата приема обучаващо изображение, прекарва го през слоеве като конволюция, ReLU и обединяване, и чрез напълно свързания слой оценява принадлежността му към („изходните вероятности“ за) както за съответния клас, така и за останалите класове.
3: Изчисляване на общата грешка в изходния слой, сумарно по всички класове:

$$\langle error \rangle^2 = \sum (\langle target\ prob. \rangle - \langle output\ prob. \rangle)^2$$

4: Обновяване на всички параметри на филтрите използвайки изчислената грешка от предходната стъпка 3, с цел минимизирането ѝ.
5: Повтаряне на стъпки 2-4, с всички изображения от тренировъчното множество, до достигане на дадено ниво на общата грешка.

3. Най-успешните CNNs напоследък

CNNs бележат бързото си развитие през 2012. Това е годината, когато за първи път те успяват да спечелят състезанието ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [5]. ImageNet съхранява около 15 милиона анотирани изображения, в общо над 22 000 категории. Главният критерий за успех в ILSVRC е, колко добре една архитектура се справя в задачи като класификация и локализация на обект в милиони от изображения. Предизвикателството ILSVRC се провежда ежегодно от 2010 г. насам, привличайки участието на повече от 50 институции до сега. Може да се смята, че това е ежегодната олимпиада по Компютърно зрение.

Победителите в тази олимпиада, както и други важни разработки са описани тук, хронологично и достатъчно детайлно за разбирането им (според достъпните източници), след което са резюмирани в сравнителен аспект в секция 4.

Посочените тук проценти за грешки в резултатите от ImageNet са главно от типа (Top-5), което означава, че при дадено входно изображение, алгоритмът/моделът не успява да възпроизведе правилната класификация на обектите в него в най-добрите 5 прогнози. Аналогично, (Top-1) означава, всеки отговор да е точен.

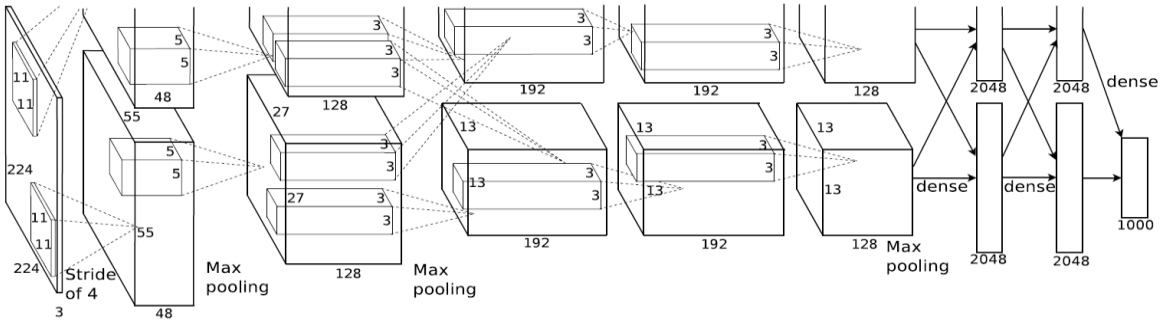
3.1. AlexNet (2012)

Публикацията [6], озаглавена „ImageNet класификация чрез CNNs“, е цитирана вече над 26 хил. пъти (според scholar.google.com) и има всички шансове да остане "ever green" в областта. Alex Krizhevsky, Ilya Sutskever и Geoffrey Hinton създават „Голяма, дълбока CNN“, която печели надпреварата ILSVRC'2012. Това е първия път, в който CNN успява да постигне 15.3% грешка (Top-5). Алгоритъмът, класирал се втори за същата година, не е от CNN тип и изостава с грешка 26.2% (Top-5). Това е изумителното постижение на CNN, шокирало общността на компютърното зрение. От тогава CNN мрежите са едни от най-използваните алгоритми в състезанието и досега.

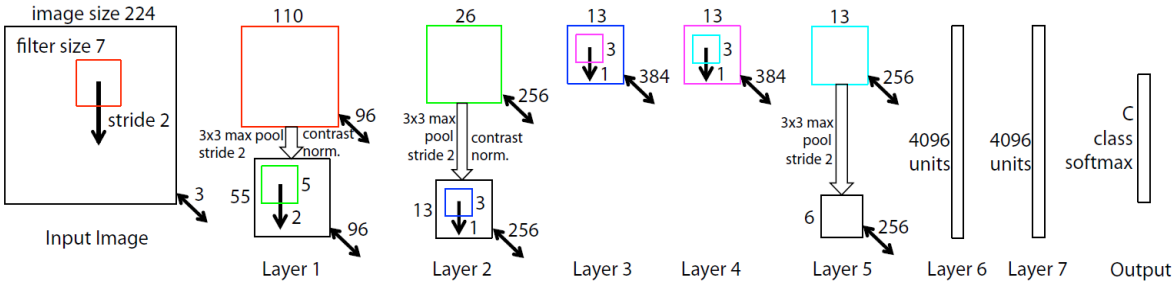
CNN архитектурата AlexNet е илюстрирана на Фиг.10. Архитектурата е относително проста, в сравнение със сегашните CNN модели, но в нея са внедрени ~650 хил. неврона, управлявани от ~60 милиона параметри (тегла). Мрежата е организирана в 5 междинни слоя (конволюции и обединения) и 3 напълно свързани слоя с финален софтмах за 1000 изхода.

Характерни особености на AlexNet:

- Мрежата е тренирана върху ImageNet, за 1000 класа;
- За нелинейна функция е използвана ReLU (софтплюс);
- За редуциране на преобучението в напълно свързаните слоеве е даден нов регуляризиращ метод "dropout";
- Използвани са техники за синтетично увеличаване на данните, чрез транслации, извличане на парчета и създаване на огледални изображения;
- Обучението на мрежата е по алгоритъма Stochastic gradient descent, със специфични стойности на параметрите за инерция и разпадане на теглата;
- Обучението е осъществено чрез два графични процесора GTX 580, за 5-6 дни.



Фиг. 10. Архитектурата на AlexNet (взаимствано от [6]). Виждат се два различни потока, защото тренировъчния процес се оказал до такава степен изчислително скъп, че се е наложило разделяне на тренирането на мрежата върху 2 различни графични процесора.



Фиг. 11. Архитектура на ZF Net (заимствано от [7]).

3.2. ZF Net (2013)

След еуфорията от AlexNet през 2012 г., броят на CNN моделите представени на ILSVRC'2013 нараства значително. Победителят в състезанието през 2013 година е мрежата ZF Net, [7] на Matthew Zeiler и Rob Fergus от Ню Йоркския университет. ZF Net успява да постигне грешка от 11.2% (Top-5). Тази архитектура (Фиг.11) е по-скоро прецизна настройка на AlexNet (виж Фиг.10), но дава нови ключови идеи за подобряване на производителността на CNN. Друга причина за популярността на [7] е, че авторите отделят особено внимание на обяснения за функционирането на CNNs и показват, как правилно да бъдат визуализирани филтрите и теглата на мрежата. Те лансират идеята, че подновеният интерес към CNNs се дължи на факта, че сега изследователите могат да използват големи обеми от тренировъчни данни, както и увеличена изчислителна мощност (графичните ускорители). Авторите споменават и за ограниченото знание за CNN, а именно относно вътрешните механизми на мрежите. Повечето разработки са сведени до „проба – грешка“. Въпреки напредъка напоследък, този проблем е все още налице.

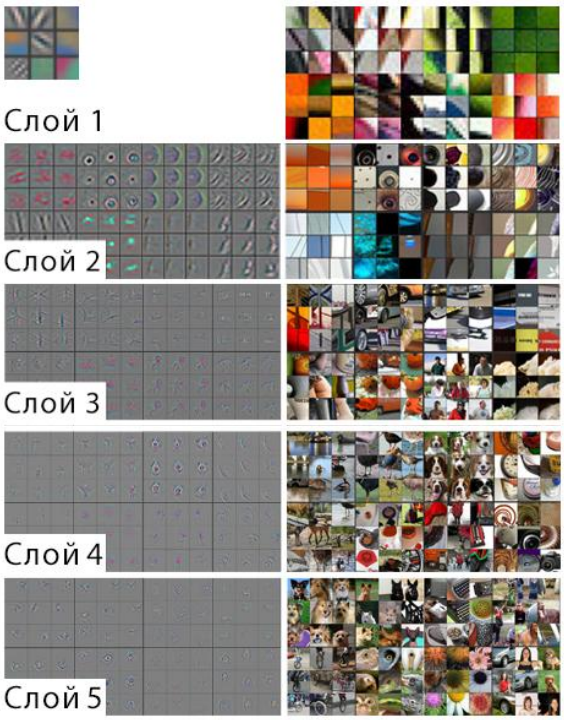
Характерни особености на ZF Net:

- Много близка до AlexNet, с изключение на някои малки (но съществени) модификации;
- ZF Net е обучавана само с 1.3 милиона изображения докато AlexNet – с 15 милиона;
- Вместо използването на филтри с размер 11×11 в първия слой на мрежата (което виждаме в AlexNet), ZF Net използва филтри с размер 7×7 и по-малко отместване при филтрирането. Идеята тук е, че използването на по-малък филтър в първия слой се съхранява съществена информация за входните данни, което се демонстрира експериментално.
- С нарастване на номера на слоя, мрежата използва повече на брой филтри;
- За активация на невроните се използва ReLU, а критерият за минимизиране на грешката е Cross-entropy loss, но самата мрежа е тренирана (подобно AlexNet) чрез пакетен (batch) режим на алгоритъма „Stochastic gradient descent“;
- Мрежата е тренирана на един графичен ускорител GTX 580, за 12 дни;
- Разработена е техниката „*деконволюционна мрежа*“ (DeconvNet) за визуализация на картите на характеристиките по слоеве, което спомага изследването на зависимостите им назад към входните данни. Названието акцентира върху съществената тук инверсия на операциите, реализирани от съответният конволюционен слой.

3.2.1. Деконволюционна мрежа (DeconvNet)

Деконволюционната мрежа се състои от определен брой деконволюционни слоеве. На всеки конволюционен слой от CNN се „закача“ *деконволюционен слой* и така, в крайна сметка се стига до пикселите на входното изображение. Това позволява да се изследва, на кои пиксели от входното изображение реагират отделните карти на характеристиките. Деконволюционната мрежа използва същите филтри като оригиналната (правата) CNN, като създава поредица от слоеве обратни на слоевете на правата CNN, до достигане на входа.

От илюстрацията на деконволюцията (Фиг. 12) се вижда, че първия слой от CNN винаги намира характеристики от ниско ниво, като открива единствено прости ръбове и определени цветове. Във втория слой обаче, се наблюдава откриване на по-сложни кръговидни структури, а в слоеве 3, 4 и 5 характеристики от по-високи нива, като глави на кучета, цветя и др..

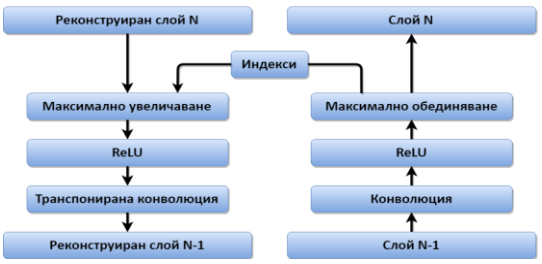


Фиг. 12. Визуализация на слой 1, 2, 3, 4 и 5 (модифицирано заимствана от [7]). Вляво (по слоеве) са визуализирани филтрите, а вдясно са показани части от тренировъчните изображения, най-силно активирани от съответния филтър.

3.2.2. Деконволюционен слой

Този слой е обратен на стандартната работа на един конволюционен слой в CNN. В представената архитектура (Фиг.13) след всеки конволюционен слой се извършва операция ReLU, след която максимално обединяване.

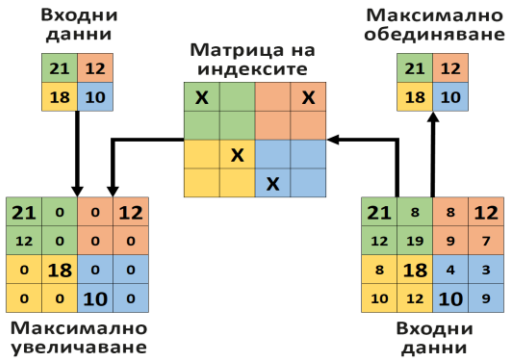
Обратното на тази последователност, наречено деконволюционен слой, се осъществява чрез операциите – *максимално увеличаване*, ReLU и *транспонирана конволюция*. Деконволюционната мрежа (от реконструирания слой и до началото на CNN) ще реконструира близка версия на слой *N*-1, *N*-2,... и така ще продължи до достигане на входното изображение. Така, можем да определим, кои входни пиксели активират слой *N*.



Фиг. 13. Деконволюционен слой (вляво) „закачен“ към стандартен конволюционен слой в CNN (вдясно).

3.2.3.Максимално увеличаване

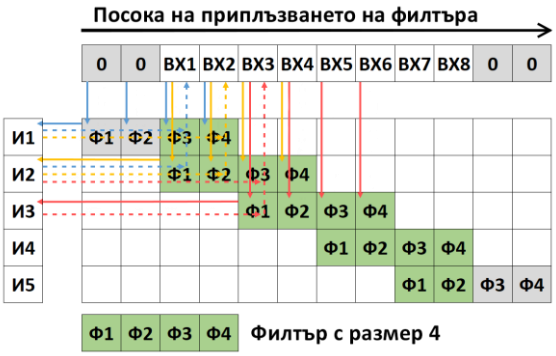
CNN операцията максимално обединяване (виж секция 2.1.3) е необратима, но ако се запазят индексите на максималните стойности, които максималното обединяване избира, може да се достигне до близка версия на обратна операция. В деконволюционната мрежа максималното увеличаване прави точно това и така запазва структурата на данните (виж Фиг.14).



Фиг. 14. Операция максимално увеличаване (ляво), използваща матрица на индексите и съответната ѝ операция максимално обединяване (дясно).

3.2.4. Транспонирана конволюция

Транспонираната конволюция е операция обратна на стандартната конволюция и може да бъде обяснена интуитивно чрез едномерни данни. Стандартната конволюция на едномерни данни е илюстрирана на Фиг.15: филтър с размер 4 се приплъзва със стъпка 2 върху входни данни с размер 8 (разширен до 12 с нули).



Фиг. 15. Конволюция на едномерни данни с филтър 1×4 и приплъзване със стъпка 2 ("плътни" стрелки). Обръщане на посоката на конволюцията ("пунктирани" стрелки).

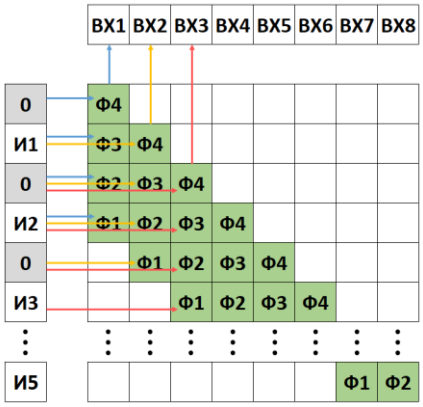
Филтърната матрица $[Ф1, Ф2, Ф3, Ф4]$ приплъзва хоризонтално със стъпка 2 по входните данни $[0, 0, ВХ1, \ldots, ВХ8, 0, 0]$, а резултатът $[И1, И2, И3, И4, И5]$ се получава (по дефиниция), като филтърът се умножава поелементно с входните данни и сумира. Стрелките във фигурата показват кои елементи от входните данни се използват за изчислението на един изходен елемент. Така:

$$И1 = (0 * Ф1 + 0 * Ф2 + ВХ1 * Ф3 + ВХ2 * Ф4)$$ и т.н.

След като знаем, че всеки изходен елемент в горната конволюция зависи единствено от 4 входни елемента, то обратното също е вярно, т.е. всеки един елемент от $[И1, И2, И3, И4, И5]$ влияе единствено върху 4 (последователни) елемента от $[0, 0, ВХ1, \ldots, ВХ8, 0, 0]$. Но, ако обърнем стрелките във Фиг.15 виждаме, че ВХ1 зависи единствено от елементите И1 и И2, ВХ2 – от И1 и И2, ВХ3 – от И2 и И3 и т.н.. Тоест, всеки един от изходните елементи зависи единствено от 2 входни елемента (което е различно от стандартна конволюция). Така при правата операция (конволюцията) всеки изходен елемент зависи от 4 входни.

При това, Ф1 и Ф3 участват в изчислението на входните елементи с нечетен индекс, докато Ф2 и Ф4 на тези с четен индекс. Това показва, че операцията обратна на конволюцията, използва 2 различни филтъра, което е много неефективно, защото за получаване на изходния резултат ще е необходима постоянната им смяна.

Но, ако между всеки два входни елемента поставим нула и транспонираме филтъра (виж Фиг.16), то връзките между входните и изходните елементи не се нарушават, а изчисленията вече се осъществяват с един и същи филтър.

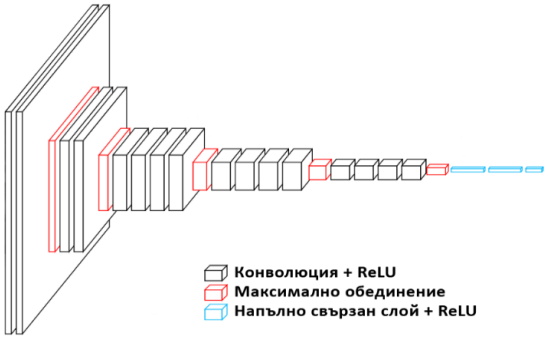


Фиг. 16. Транспонирана конволюцията от Фиг.17.

Така, транспонираната конволюция може да бъде извършена по същия начин като стандартната конволюция, но след равномерно разширяване на входните елементи с нули и транспониране на оригиналния филтър.

3.3. VGG Net (2014)

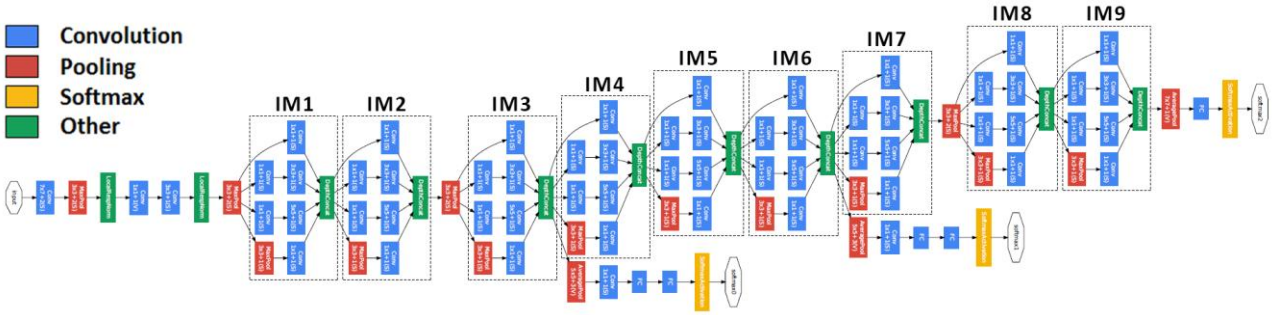
Това е CNN моделът [8] създаден през 2014 година, който не е победител на ILSVRC'2014, но показва колко важно е една CNN да бъде с проста архитектура и голяма дълбочина. VGG Net постига грешка от 7.3% (Top-5). Karen Simonyan и Andrew Zisserman от университета в Оксфорд създават тази CNN, която стриктно използва филтри с големина 3×3, стъпка на отместване 1 и разширяване с 1, заедно с 2×2 обединяващ слой с отместване 2. Авторите създават две разновидности на мрежата, съответно VGG16 и VGG19 (Фиг.17), като числата 16 и 19 указват, колко конволюционни и напълно свързани слоя съдържа всяка от тях.



Фиг. 17. Архитектура на VGG19.

Характерни особености на VGG Net:

- Използват се само 3×3 филтри, за разлика от филтрите 11×11 в AlexNet и 7×7 в ZF Net. Аргументът на авторите е, че комбинацията от два конволюционни слоя 3×3 има поле на възприятие 5×5. Така чрез малки по големина филтри може да се симулира по-голям филтър. Предимството е по-малък брой параметри в мрежата. Освен това, с два конволюционни слоя могат да се използват и два ReLU слоя вместо един.
- Тъй като пространственият размер на входните (междинните) данни след всеки слой намалява (в резултат от конволюция и обединяване), броят на характеристичните карти расте, което спомага за откриване на повече типове характеристики.



Фиг. 18. Архитектура на GoogLeNet, съставена от 9 основаващи модула (модифицирано, заимствано от [9]).

- Броят на филтрите се удвоява след всеки обединяващ слой, което засилва ефекта от стесняването на пространствените размери и увеличаване на дълбочината;
- Моделът работи добре, както при класифициране на изображенията, така и при откриване на обекти в тях;
- Използвани са техники за синтетично увеличаване на входните данни на етапа на обучението (тренирането) на мрежата;
- Използва се ReLU след всеки конволюционен слой;
- Алгоритъмът за обучение е от типа Mini-batch gradient descent;
- Мрежата е обучавана на 4 графични ускорителя "nVidia Titan Black", за около 2-3 седмици.

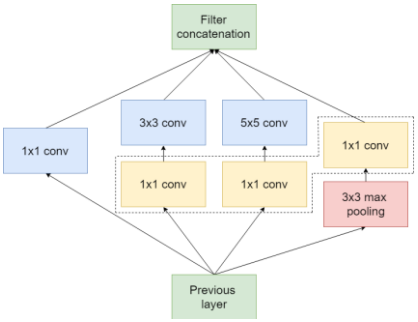
3.4. GoogLeNet (2014)

През същата 2014 година се появява още една интересна CNN архитектура, коренно противоположна на VGGNet. Това е мрежата GoogLeNet [9], на Гугъл, победител в ILSVRC'2014, с грешка 6.7% (Top-5). GoogLeNet (виж Фиг.18), представя т.нар. *„основаващ“ (inception)* модул, показан на Фиг.19. Това е една от първите архитектури, които отклоняват CNN от традиционното последователно прилагане на конволюционни и обединяващи слоеве върху входните данни. Авторите поясняват, че с новата архитектура акцентират върху използването на паметта, а също на процесорите и графичните ресурси.

3.4.1.„Основаващ“ (Inception) модул

За разлика от разгледаните дотук CNNs, където процесите се извършват последователно, в GoogLeNet определени операции се изпълняват паралелно.

Във всеки слой се извършват паралелни операции от различни типове конволюция и обединение, както и избор, какъв да е размерът на филтрите, за да се извлича "по-фина" информация. Основаващият модул позволява всичко това да се извършва паралелно. Всъщност това е била първоначалната идея, илюстрирана на Фиг.19.



Фиг. 19. „Основаващ“ (Inception) модул (модифицирано заимствано, [9]). В първоначалната версия на модула са липсвали 3-те 1x1 конволюции (оградени с пунктир, тук).

Но, първоначалният замисъл не е проработил, защото е създавал твърде дълбоки карти на характеристиките. Начина, по който авторите се справили с това, е добавянето на т.нар. *единична 1×1 конволюция* преди всеки 3×3 или 5×5 слой, (на Фиг.19, добавката е оградена с пунктир).

Единичната конволюция, използвана тук, предоставя метод за намаляване на размерите. Примерно, ако имаме входни данни с размер 100×100×60 и на тях приложим 20 филтъра с размер 1×1×60, резултатът ще има размер 100×100×20. Това означава, че конволюциите с размер 3×3 и 5×5 ще боравят с по-малък по обем данни. Този подход може да се смята за обединяващ характеристиките, защото намалява дълбочината на изходните данни.

Прочее, "inception" идеята е доразвита през 2015г., за постигането на още по-фрапиращи резултати в редукцията на памет и на процесорно време, например, Inception-v3, [15], Inception-v4 и Inception-ResNet, [16].

Архитектурата GoogLeNet представлява (най-общо) поредица от основаващи модули (общо 9 броя). Всеки модул може да извлича специфична детайлна информация от входните данни, както и да обхваща голяма част от тях. При това, модулът извършва операция обединяване, която намалява размера на данните, като същевременно подпомага срещу проблема за претренирането на модела в цялост. Финално се извършва и ReLU (софтмах) операция, която допринася за нелинейността на мрежата. Така GoogLeNet е в състояние да изпълнява всички тези операции, като в същото време не изисква огромна изчислителна мощ.

Характерни особености на GoogLeNet:

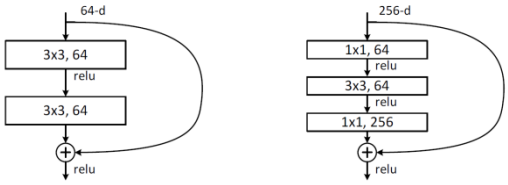
- Архитектурата на мрежата съдържа главно 9 основаващи *(inception)* модула;
- Не се използват напълно свързани слоеве. За преминаване от размер 7×7×1024 към 1×1×1024 се използва осреднено обединяване. Това спестява голям брой параметри;
- Използва 12 пъти по-малко параметри от AlexNet;
- Вместо входното изображение, на мрежата се подават известен брой негови отсичания (image crops), т.е. достатъчно информативни версии, а крайният резултат се определя като средно аритметично по версиите;
- Обучението е извършено за 1 седмица на графични ускорители от висок клас.

3.5. Microsoft ResNet (2015)

ResNet [10] е CNN архитектура състояща се от 152 слоя, създадена от Майкрософт, която поставя нови рекорди в класификацията на изображения и локализирането на обекти в тях. Тя печели ILSVRC'2015 с грешка от 3.6% (Top-5). Нека припомним, че при решение на същите задачи, хората постигат грешка от порядъка на 5%, [5]. Новостта, която ResNet представя е т.нар. *„остатъчен“ (residual) блок*, показан на Фиг.20.

3.5.1.Остатъчен (residual) блок

Идеята на този блок е, входните данни да преминават през серия от конволюционни, обединяващи и ReLU слоеве, като крайния резултат от тази серия да се сумира с оригиналните входни данни, и така променяйки ги подходящо. Тоест, вместо да се изчислява някаква трансформация на входните данни, остатъчният блок изчислява подходящ адитивен терм към тях, т.е. някакво отклонение на входните данни, а не напълно нова тяхна репрезентация, както мрежите разгледани досега.



Фиг. 20. Остатъчен *(Residual)* блок: две различни интерпретации (заимствано от [10]).

Причина за наблюдаваната ефективност на остатъчния блок е това, че по време на обучението "backpropagation", градиентът се променя по-плавно от слой към слой, поради допълнителните операции на блока, разпределящи градиента. Прочее, "residual" идеята е комбинирана успешно с "inception", 2015г., виж Inception-ResNet, [16].

По отношение дълбочината на мрежата, авторите твърдят, че „наивното“ увеличаване на броя на слоевете в CNNs не води до очакваното понижение на изходната грешка, но това не важи за подхода *"residual* обучение". От друга страна, достигайки до мрежа с 1202 слоя, те са получили по-малка точност върху тестовите данни, но обясняват това с претренирането на мрежата върху същите обучаващи данни (които вече се оказват недостатъчни по обем).

Характерни особености на ResNet:

- ResNet е изградена като контрапункт на VGG-19. Сложността ѝ (в GFlops) е ~5 пъти по-малка от тази на VGG, въпреки че ResNet има ~8 пъти повече слоеве от VGG. Но, след 2-я слой, размерът на входящите данни (по слоеве) „драстично“ спада от 224×224 на 56×56;
- Мрежата използва съществено иновацията „остатъчно *(residual)* обучение“, като формира съответните блокове върху болшинството от двойките (или тройките) последователни конволюции (по слоеве);
- Обучението е отнело от 2 до 3 седмици на 8 графични ускорителя.

3.6. SegNet (2016)

SegNet [11] е CNN, насочена към семантичния анализ на сцени чрез регионално сегментиране на ниво пиксели, в определен брой класове. SegNet определя класа на принадлежност за всеки пиксел от входното изображение. Де-факто, мрежата представлява комбинация от 2 две мрежи, *кодираща* (encoding) и *декодираща* (decoding) (Фиг.21). Архитектурата на кодиращата мрежа е топологично идентична с мрежата VGG16 до 13-тия ѝ конволюционен слой. Ролята на декодиращата е да осъществи връзка между картите на характеристиките в кодиращата мрежа, а именно – между картите с ниска резолюция и картите с резолюция, равна на входната.

Тази архитектура показва, че пропускането на напълно свързания слой не води до намаляване на точността на една мрежа, а в същото време се намалява значително броя на параметрите в нея. Така, броят на параметрите на SegNet е намалял от 134 милиона докъм 14.7 милиона (т.е. ~ 9 пъти) в сравнение с други архитектури за семантичен анализ, а именно Fully CNNs, [12] и de-CNNs, [13].

Фиг. 21. Архитектура на SegNet (заимствано от [11]).

В SegNet, всеки *кодиращ слой* се състои от *n* на брой конволюции, следвани от същия, *n* на брой ReLU, завършващи с операция максимално обединение. На всеки кодиращ слой съответства *декодиращ слой*, който извършва точно обратните операции. Декодиращите слоеве се състоят от максимално увеличаване, следвано от *n* на брой конволюции. Операцията максимално увеличаване използва индексите на максималните стойности от съответната операция максимално обединение (от кодиращия слой), като по този начин се реконструират по-точно "стимулите" в изображението, които влияят на сегментацията.

Характерни особености:

- SegNet съхранява само индексите за макс-обединяване на картите на характеристиките и ги използва в своята декодираща част, за да постигне добри резултати.
- Значително по-малкия брой на параметри за обучаване води до ефективност, както по памет, така и по изчислително време, в сравнение с други архитектури, напр. VGG-16 [8], а също тези от [12] и [13].
- SegNet е разработена главно за т.нар. „дълбоко сегментиране“, анализ, разбиране на транспортни (пътни) сцени, както и интериорни (вътрешни) сцени;
- Обучението е от типа „от начало до край“ (end-to-end), което си е предизвикателство, но е успешно решено по метода Stochastic gradient descent.

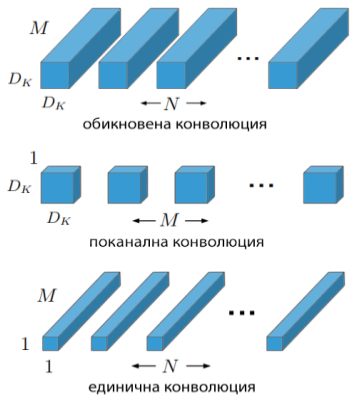
3.7. MobileNets (2017)

MobileNets е нова ефективна CNN архитектура от 2017г., представена от Google, [14]. Мрежата се конфигурира от 2 хиперпараметъра, позволяващи да бъдат създавани много малки и с ниска латентност модели, които леко да се вграждат в мобилни и/или други вградени устройства. MobileNets е публикувана в отговор на тенденцията да се създават по-дълбоки и по-сложни мрежи, целящи постигането на по-висока точност, но което прави мрежите неефикасни по отношение на размери и скорост [8, 10, 15, 16]. MobileNets са изградени главно от т.нар. „поканално единична конволюция“ (Depthwise separable convolution), представена още в [17].

3.7.1.Поканално единична конволюция

Това е конволюционен оператор даващ краен резултат еквивалентен на стандартната конволюция, но отличаващ се по начина на изчисление на параметрите. Той съдържа в себе си „*поканална конволюция*“ и „*единична конволюция*“, изпълнени последователно. Поканалната конволюция прилага филтри, поотделно на всеки един от каналите на входните данни, а единичната конволюция след това прилага 1×1 конволюция върху резултата, за да ги обедини (виж Фиг.22).

Тази техника на разделяне на стандартната конволюция на 2 етапа, филтриране и комбиниране, драстично намалява крайния размер на модела, както редуцира и обема на изчисленията в него. В поканално единичната конволюция след всяка от операциите се прилагат *нормализиращ* слой и ReLU слой.



Фиг. 22. Поканално единична конволюция (заимствано от [14]).

Като използват 3×3 поканална единична конволюция, MobileNet мрежите редуцират броя на изчисленията 8-9 пъти, с цената на едно незначително понижаване на точността. Архитектурата им се състои от 28 слоя, като започва със стандартен конволюционен слой, последван от слоеве поканално единични конволюции и завършва със слой осреднено обединяване, последван от напълно свързан слой. Тези мрежи са почти толкова точни, колкото VGG16, но са 32 пъти по-малки по обем и с 27 пъти по-кратки изчисления. А са по-точни от GoogleNet, като отново са по-малки по-обем и с 2.5 пъти по-кратки изчисления (виж Табл.1 от следващата секция 4).

4. Сравнителен анализ на разгледаните CNNs.

В предходната секция бяха описани в хронологичен порядък основните архитектури на CNNs, свързани с бурното им развитие напоследък. Особено внимание бе отделено на тези иновации в CNNs, които допринасят съществени идеи за развитие в следващите ги CNNs, независимо от факта, че всички са свързани най-вече с класификацията на изображения. Тук ще систематизираме написаното, сбито в таблица (Табл. 1), която да спомогне за лесното проследяване на еволюцията на тези мрежи спрямо точността им на разпознаване на обекти, както и нововъведенията в структурен план, които те представят.

Таблицата е организирана хронологично, според колоната „Година”, в която съответната CNN с даденото „Име” е победила на ежегодното състезание „Top 5 of ImageNet”. За ориентация е дадено и съответното ниво на грешка от типа (Top-1), също в проценти. Главна цел на таблицата е да резюмира (и да разграничи) основните „Нововъведения”, характерни за разгледаните CNNs. Същевременно, подчертава се и тенденцията, че с всяка изминала година съответната класификационна „Грешка” става все по-малка и вече е съизмерима и/или доминира (виж ResNet) средната човешка грешка (~5%) при класификация на изображения, [5]. Напоследък фокусът на изследванията се измества към намаляване на обема (общия брой) на параметрите на CNNs, виж MobileNets [14], където това е акцент и се постига с цената дори на известно влошаване на класификационната грешка.

Относно дълбочината на мрежите, т.е. броят на слоевете в разгледаните CNN архитектури, смятаме че тази характеристика се определя главно от структурата на входните данни (които тук са 2D изображения) и можем да говорим за едно стабилизиране в рамките на ~20-ина слоя. Колкото до дълбочината на всеки от слоевете, т.е. броят на филтрите, респ. броят на характеристичните карти на слой, то те варират в доста широки граници, което е в пряка зависимост и от предвижданите (възможните) приложния на мрежата. Естествен е стремежът напоследък към редукция в сложността, с което се съкращава времето за изпълнение и така се „отварят вратите” към множество нови приложения на CNNs, например в ежедневието, където компютърните мощности традиционно са по-ограничени.

5. Заключение

Конволюционните невронни мрежи (CNNs) се различават от множеството други ANN архитектури по това, че CNNs използват информация за типа на получаваните входни данни (напр., изображение с дадени размери). Последното позволява опростявания на мрежовата им архитектура, която и без друго е принципно усложнена. Но, независимо от засиления интерес и огромния брой проведени изследвания, основните принципи за изграждането на невронни мрежи остават непроменени и в CNNs, засега.

Таблица 1: Сравнителен анализ на архитектурите на CNNs представени в секция 3.

Име	Година	Грешка в ImageNet		Параметри (милиони)	Брой слоеве	Нововъведения
		(Top-5)	(Top-1)			
AlexNet [6]	2012	15.3%	37.5%	60	8	Първата CNN, (Top-5) победител в състезанието ILSVRC.
ZF Net [7]	2013	11.2%	n/a	62	8	Яснота във функционирането на CNN; DeconvNet визуализация и настройка; Деконволюц. слоеве; Транспонирана конволюция; Max увеличаване;
VGG NET [8]	2014	7.3%	28.5%	138	16 / 19	Акцент върху дълбочината на CNN; Редуциране на размера на филтрите
GoogLeNet [9]	2014	6.7%	30.2%	6.8	22	Не е задължително CNN слоевете да работят последователно; "Основаващ" (Inseption) модул.
ResNet [10]	2015	3.6%	19.4%	60	152	Рекордно ниска грешка 3.6% (Top-5); "Остатъчен" (Residual) блок.
SegNet [11]	2016	n/a	n/a	14.7	27	CNN към семантичната сегментация на изображения; Игнорира напълно свързания слой.
MobileNets [14]	2017	10.5%	29.4%	4.2	28	CNN за малки устройства с нископроизводителен хардуер; Поканално единична конволюция (Depthwise separable convolution).

Настоящата статия представя основните принципи на функциониране и значителния напредък, постигнат при проектирането на различни архитектури на CNNs, като разглежда основните модули при изграждането на слоевете в дълбочина на една пълна и завършена мрежа. Целта е да се разсее първоначалната неяснота на човек, който има интерес към такъв тип невронни мрежи и същевременно, да се направи по-достъпна за начинаещия, както идеята „CNNs“, така и областта Компютърно зрение, където CNNs направиха пробив напоследък, а защо не и да се провокира търсенето и на други възможни техни приложения.

След придобиване на необходимите основни знания, тук читателят може да се запознае с еволюцията на архитектурите на CNNs. Обобщени са най-значимите публикации в областта, показващи пъстротата на публикуваните CNNs през последните години.

Някои подробности около изграждането на мрежите и изчисленията „слой по слой“ са опростени или пропуснати с цел читателят да схване интуитивно и по-цялостно картината на CNNs, като същност и развитие. Същевременно, направен е опит за прецизен (най-често контекстен) превод на български език на новите понятия тук, но са съхранени повечето оригинални английски абревиатури, напр. CNNs. Възможно е този превод/подход да породи езикова дискусия, затова някои „критични“ термини използваме дублирано (български/английски).

Главните резултати на този обзор са хронологично представени в секция 3, където е дадено подробно описание на разгледаните архитектури, а секция 4 съдържа сбит сравнителен анализ.

Надяваме се, обзорът да осигури не само по-добро разбиране на конволюционните невронни мрежи, но и да улесни бъдещите изследователски дейности, както и развитието на приложенията в областта.

Разбира се, за едно по-пълно разбиране на алгоритмите и интуицията, които стоят зад обсъжданите архитектури, е препоръчително да се прегледат по-задълбочено оригиналните литературни източници, представящи всяка една от тях.

Списък на цитираната литература

[1] Masters T.: Signal and image processing with neural networks, John Wiley & Sons, Inc., NY, 1993.

[2] Hubel, D. H., T. N. Wiesel: Receptive fields and functional architecture of monkey striate cortex. Journal of Physiology, London, 1968, pp. 215-243.

[3] Hochreiter, J.: Untersuchungen zu Dynamischen Neuronalen Netzen. Diplomarbeit, Institut fur Informatik, Technische Universitat Munchen, 1991.

[4] Nair, V., G. E. Hinton: Rectified Linear Units Improve Restricted Boltzmann Machines. ICML, Haifa, Israel, 2010, pp. 807-814.

[5] Russakovsky, O., J. Deng, H. Su, J. Krause, S. Sanjeev, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, F.F. Li: ImageNet Large Scale Visual Recognition Challenge. IJCV, 2015.

[6] Krizhevsky, A., I. Sutskever, G. E. Hinton: Imagenet Classification with deep convolutional neural networks. Advance in NIPS, 25, 2012.

[7] Zeiler, M. D., R. Fergus: Visualizing and understanding convolutional networks. ECCV 2014, Springer, LNCS 8689, Part I, pp. 818–833, 2014.

[8] Simonyan, K., A. Zisserman: Very deep convolutional networks for large-scale image recognition. ICLR, 2015.

[9] Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich: Going deeper with convolutions. CVPR, pp.1-9, 2015.

[10] He, K., X. Zhang, S. Ren, J. Sun: Deep residual learning for image recognition. CVPR, 2016.

[11] Badrinarayanan V., A. Kendall, R. Cipolla: SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. PAMI, 2017.

[12] Long, J., E. Shelhamer, T. Darrell: Fully convolutional networks for semantic segmentation. CVPR, pp. 3431–3440, 2015.

[13] Noh, H., S. Hong, B. Han: Learning deconvolution network for semantic segmentation. ICCV, pp. 1520–1528, 2015.

[14] Howard, A.G., M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam: Mobilenets: Efficient convolutional neural networks for mobile vision applications. CoRR, 2017.

[15] Szegedy, C., V. Vanhoucke, S. Loffe, J. Shlens, Z. Wojna: Rethinking the inception architecture for computer vision. CVPR, 2016.

[16] Szegedy, C., S. Loffe, V. Vanhoucke: Inception-v4, inception-resnet and the impact of residual connections on learning. AAAI, 2017.

[17] Sifre, L.: Rigid-motion scattering for image classification. Ph. D. thesis, Ecole Polytechnique, CMAP, 2014.

За контакти:

Инж. **Антон Христов**[1]
e-mail: anhristov@gmail.com
тел. 0879 989 850

Проф. **Мария Нишева**[1]
e-mail: marian@fmi.uni-sofia.bg
тел. 0879 243 434

Доц. **Димо Димов**[1,2]
e-mail: dtdim@iinf.bas.bg
тел. 0884 309 548

[1] Факултет по математика и информатика – СУ „Св. Климент Охридски“

[2] Институт по информационни и комуникационни технологии – БАН