Варненски свободен университет "Черноризец Храбър"

Факултет „Социални, стопански и компютърни науки“

Катедра „Компютърни науки“

Дипломна работа

Тема:

"Анализ на астрономически изображения чрез симетрични преобразувания в невронни мрежи"

"Symmetric transformations in neural networks applied to astronomical images"

Дипломант:

Йоана Фотева

Научен ръководител: доц. д-р Стоян Мишев

2024

Съдържание

Списък на фигурите

Глава 1

Увод

С развитието на съвременните технологии изкуствения интелект и машинното обучение стават все по-важни инструменти в научните изследвания, а усъвършенстването им води до по-точни и надеждни резултати. [1] В областта на астрофизиката и астрономията количеството и сложността на данните нарастват с всеки нов проект и разработването на ефектвни модели, които да се спряват успешно с тези обеми от данни, е от ключово значение. Особено предизвикателство представлява задачата за разпознаване и класификация на астрономически обекти въз основа на техните изображения. Традиционните подходи, които не отчитат сложните симетрии в изображенията, често се оказват недостатъчно ефективни или точни, което подчератава нуждата от иновативни решения. [3]

С тази дипломна работа ще бъде представено решение за ана-

лиз на астрономически изображения чрез използването на еквивариантна невронна мрежа (E2CNN), която притежава способността да разпознава и обработва симетрии като ротации и транслации, запазвайки ключовата информация, съдържаща се в данните. Тази работа ще акцентира върху значението на симетрията в обработката на данни и как еквивариантните невронни мрежи предоставят нови възможности за анализ на изображения.

За целите на изследването е използвана библиотеката `escnn` в PyTorch и набора от данни на Galaxy Zoo. Изследването включва както теоретични основи, така и практическа част за изграждането и тестването на еквивариантна невронна мрежа, като по този начин предлага важен принос в областта на астрофизиката и машинното обучение.

1.1 Цел

В тази дипломна работа ще разгледаме разработването на напреднали модели на невронни мрежи, които чрез интегриране на симетрични трансформации, подобряват анализа на астрономическите изображения. Основната цел е да се адресират предизвикателствата, по-специално на ротационната и транслационната инвариантност на изображенията на галактики, които могат значително да повлияят на производителността на традиционните конволюционни невронни

мрежи (CNN).

1.2 Ползи от дипломната работа

Тази дипломна работа допринася за подобряването на съвременните методи за анализ на астрономически изображения чрез интегриране на еквивариантни невронни мрежи. Този подход е особено важен в условията на постоянно увеличаващото се количество данни в съвременната астрофизика и астрономия, което може да допринесе за нови открития и по-задълбочено разбиране на Вселената.

1.3 Структура на дипломната работа

Дипломната работа е организирана по следния начин:

1. **Теоретични основи**

   В тази част от дипломната работа ще разгледаме основните концепции в теорията на групите, които са от съществено значение за разбирането на еквивариантните невронни мрежи. Ще се обърне внимание на определението за група, както и ще се разгледат групата на транслациите $(\mathbb{R}^2, +)$ и рото-транслационната група $SE(2)$, които моделират симетрии в двумерното пространство.

След това ще се фокусираме върху действията на групи и представяния, включително как групите могат да действат на различни пространства. Също така ще обърнем внимание на неприводимите представяния, които са ключови за анализа на симетрии. Тази теоретична основа е важна за по-дълбокото разглеждане на еквивариантните невронни мрежи в следващата част от работата.

2. **Еквивариантни невронни мрежи**

   Във втората част на дипломната работа ще разгледаме концепцията за еквивариантни невронни мрежи, които интегрират групови симетрии в своите архитектури. Първо, ще разгледаме груповите конволюционни невронни мрежи (G-CNNs) и ще анализираме различни аспекти на тяхната конструкция, включително кръстосаната и повдигаща корелация, както и методите за изграждане на тези мрежи.

   След това ще разгледаме еквивариантните преобразувания спрямо групата на Евклид, които са важни за разбирането на симетричните свойства на мрежите. В заключение, ще обърнем внимание на управляемите ядра и тяхното значение за постигане на еквивариантност в невронните мрежи.

3. **Модел на еквивариантна невронна мрежа**

   В третата част ще се представи как е приложен еквивариан-

тен невронен модел, предназначен за анализ на астрономически изображения. Първо, ще се направи преглед на набора от данни, който ще бъде използван в изследването. След това ще се представи имплементацията на E2CNN в PyTorch, включително въвеждането на пакета escnn, дизайна на мрежовата архитектура и обучението на модела.

Накрая, ще разгледаме експерименталната настройка и резултатите от проведените тестове, включително конфигурацията на експериментите, оценъчните метрики и анализ на получените резултати. Тази част ще предостави практическото приложение на теоретичните концепции, разгледани в предишните глави.

4. **Заключение и бъдещи изследвания**

Накрая ще обсъдим резултатите от това изследване и потенциала за бъдещи разработки в тази област. Това ще включва предложения за допълнителни подобрения на архитектурата на модела и допълнителни експерименти, които биха могли да надградят откритията.

Глава 2

Въведение в теорията на групите

Теорията на групите е основополагаща за разбирането на симетриите в различни математически дисциплини, като геометрия, алгебра и физика. В контекста на невронните мрежи, приложени за анализ на астрономически изображения, теорията на групите предоставя математическата рамка за описване и използване на присъщите симетрии на тези изображения. Тази рамка може да бъде от ключово значение за подобряване на ефективността и способността за по-добър анализ на моделите.

2.1 Определение

Теорията на групите е област в абстрактната алгебра, която изучава алгебрични структури, наречени групи. Групата се състои от множество от елементи и операция, която свързва два от тези елементи, така че да произведе нов елемент, който също принадлежи към групата [10].

Формално елементите на групата удовлетворяват следните четири условия:

- затвореност (closure): при дадени два елемента g и h от G, произведението $g \cdot h$ също е в G.
- асоциативност: за $g, h, i \in G$ произведението е асоциативно: $g \cdot (h \cdot i) = (g \cdot h) \cdot i$.
- единичен елемент: съществува единичен елемент $e \in G$ такъв, че $g = g \cdot e = g$ за всяко $g \in G$.
- обратен елемент: за всяко $g \in G$ съществува обратен елемент $g^{-1} \in G$ такъв, че $g^{-1} \cdot g = g \cdot g^{-1} = e$.

2.2 Група на транслациите $(\mathbb{R}^2, +)$

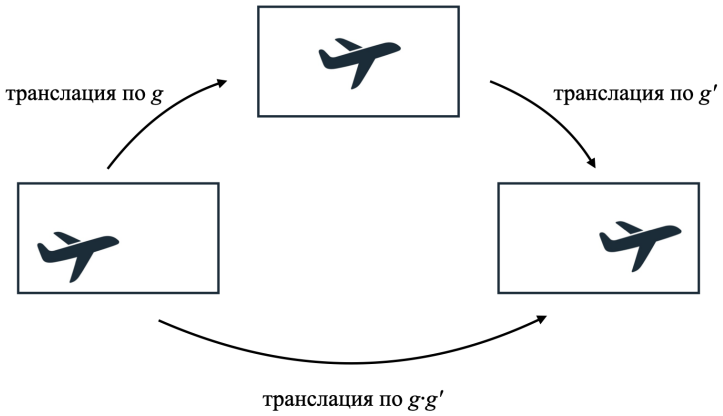
Групата на транслациите (или транслационната група) описва всички възможни премествания на обект в пространството. В тази група

имаме всички възможни транслации, параметризирани от вектори в $\mathbb{R}^2$, притежава групово произведение и групова инверсия. Можем да комбинираме две транслации чрез векторно добавяне, а обратната операция обръща посоката на вектора:

$$g \cdot g' = (x + x')$$

$$g^{-1} = (-x)$$

при $g = (x)$, $g' = (x')$ и $x, x' \in \mathbb{R}^2$.



Фигура 2.1: Илюстрация на груповата същност на операцията *транслация*.

Тази група е от съществено значение, особено когато се разглеждат преобразувания в задачи за компютърно зрение.

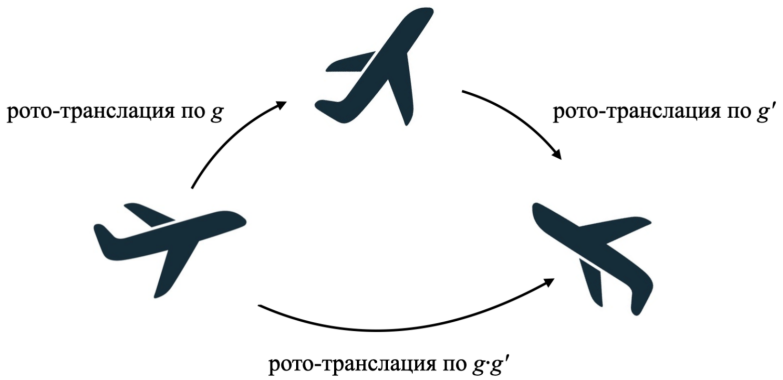
2.3 Рото-транслационна група $SE(2)$

При групата за ротации и транслации $SE(2)$ (специален евклидов) обикновеното събиране на вектори не е достатъчно.Групата $SE(2) = \mathbb{R}^2 \rtimes SO(2)$ се състои от свързаното пространство $\mathbb{R}^2 \times S^1$ на транслационни вектори в $\mathbb{R}^2$ и ротации в SO(2) (или еквивалентни ориентации в S^1) и притежава групово произведение и обратна група:

$$g \cdot g' = (x, R_\theta) \cdot (x', R_{\theta'}) = (R_\theta x' + x, R_{\theta+\theta'})$$

$$g^{-1} = \left(-R_\theta^{-1} x, R_\theta^{-1}\right)$$

при $g = (x, R_\theta)\,, g' = (x', R_{\theta'})$.



Фигура 2.2: Илюстрация на груповата същност на операцията *ротация*.

За да представим групата на ротациите и транслациите $SE(n)$ по начин, който е подходящ както за теоретични, така и за изчислителни приложения, използваме матрично представяне, при което

елемент от $SE(n)$ се изразява като $(n+1) \times (n+1)$ матрица:

$$\begin{pmatrix} R & t \\ 0^T & 1 \end{pmatrix}$$

, където:

- $R \in SO(n)$ е $n \times n$ ротационна матрица.
- $t \in \mathbb{R}^n$ е n-мерен вектор-стълб за транслация.
- O^T е вектор-ред от нули с дължина n.
- долният десен елемент е 1.

  Елемент от $SE(2)$ може да бъде представен като матрица 3×3:

$$\begin{pmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

  ,където θ е ъгълът на завъртане, а t_x и t_y са транслационните компоненти в посоките x и y. Тази матрица, когато се умножи по точка в хомогенни координати $(x, y, 1)^T$, извършва както завъртане под ъгъл θ, така и транслация по (t_x, t_y).

2.4 Действия на групи и представяния

2.4.1 Действия на групи

В този раздел ще разгледаме как преобразуванията се комбинират едно с друго. За да разширим това до прилагане на преобразувания към други обекти, като вектори или функции, трябва да въведем концепциите за групови действия и групови представяния.

Груповото действие описва как елементи от група могат да действат върху елементи от друго пространство, поддържайки структурата на групата. Групово действие върху пространство X се определя от двоичен оператор $\circ : G \times X \to X$, което удовлетворява:

$$g \odot (g' \odot x) = (g \cdot g') \odot x,$$

където $g, g' \in G$ и $x \in X$.

Това определение гарантира, че действието на прилагане на два групови елемента последователно е еквивалентно на прилагането на тяхното произведение.

2.4.2 Представяния на групи

За да приложим групови действия конкретно към векторни пространства, използваме концепцията за представяния. Представянето е линейна трансформация $\rho(g) : V \to V$ за всеки групов елемент $g \in G$, която поддържа груповата структура. С други думи, това е хомоморфизъм от групата G към общата линейна група $GL(V)$ на обратими линейни трансформации върху векторно пространство V, удовлетворяващо:

$$\rho(g)\rho(h)v = \rho(g \cdot h)v,$$

за $v \in V$.

Матрично представяне: Матричното представяне е специфичен тип представяне, което действа върху крайномерно векторно пространство $\mathbb{R}^d$, използвайки $d \times d$ матрици. Матриците $D(g)$ са параметризирани от групови елементи и се придържат към:

$$D(g)D(g')x = D(g \cdot g')x,$$

за $x \in \mathbb{R}^d$ и $D(g) \in GL(d, \mathbb{R})$.

2.4.3 Действия на групи върху функционални пространства

Важен случай на векторни пространства са т.нар. функционални пространства. Така напр. функционалното пространство $L^2(X)$ на квадратично интегрируемите функции и дадена група може да действа и върху това пространство. Един често срещан начин за представяне на това действие е чрез ляворегулярното представяне (left-regular representation).

Определение (Ляворегулярно представяне): За функция $f \in L^2(X)$ и група G, която действа върху множеството X, ляворегулярното представяне се дефинира като:

$$[\mathcal{L}_g f](x) = f\left(g^{-1} \odot x\right)$$

където $\odot$ означава действието на G върху X. Това представяне се нарича „ляворегулярно“, защото групата действа върху множеството на функцията отляво.

2.4.4 Неприводими представяния

Неприводимо представяне (irrep) е представяне, което не може да бъде разложено на по-прости, нетривиални представяния. С други думи, неприводимото представяне е най-фундаменталният

градивен елемент, от който могат да бъдат изградени по-сложни представяния.

Определение. Едно пространство е *инвариантно* спрямо дадено действие на група G, ако за всяка функция или вектор f от това пространство, приложението на всякакъв елемент g от групата G върху f води до резултат, който отново принадлежи на същото пространство. Формално, ако V е пространството и G действа върху него, то V е инвариантно спрямо G, ако за всяко $g \in G$ и $f \in V$, $g \cdot f \in V$. [7]

Определение. Представяне V на група G се нарича *неприводимо*, ако то няма инвариантни (, ненулеви) **под**пространства $W \subset V$.

В контекста на симетричните преобразувания в невронните мрежи, разбирането и използването на неприводимите представяния играе съществена роля. Например, слоеве на невронните мрежи, които запазват определени симетрии, могат да бъдат конструирани чрез представяния на групите, съответстващи на тези симетрии. Създавайки мрежи, които са еквивариантни спрямо тези симетрии, се осигурява инвариантност на произведените от модела резултати по отношение на съответните преобразувания.

Глава 3

Еквивариантни невронни мрежи

3.1 Групови конволюционни невронни мрежи (G-CNNs)

С установените теоретични основи на теорията на групите става възможно изграждането на групови конволюции в невронните мрежи. Първо ще разгледаме формулировката на конволюцията и по-специално на кръстосаната корелация.

3.1.1 Кръстосана корелация

Кръстосаната корелация (cross-correlation) се състои в изместването на ядро k спрямо функция f и изчисляването на тяхното скаларно произведение във всяка пространствена позиция. Макар термините "конволюция"и "кръстосана корелация"често да се използват взаимозаменяемо, те всъщност обозначават различни операции: при конволюцията ядрото се обръща преди прилагането му, докато при кръстосаната корелация това не се случва.

$$(k \star_{\mathbb{R}^2} f)(\mathbf{x}) = \int_{\mathbb{R}^2} k(\mathbf{x}' - \mathbf{x}) f(\mathbf{x}') \, \mathrm{d}\mathbf{x}' = (\mathscr{L}_g k, f)_{\mathbb{L}_2(\mathbb{R}^2)}$$

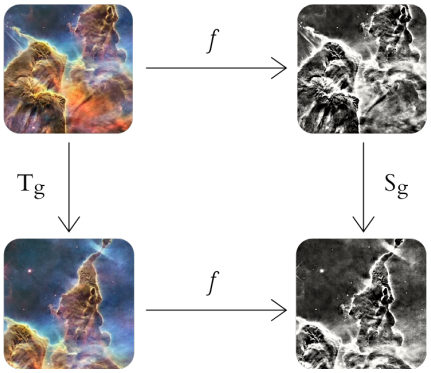
При кръстосаната корелация ядрото k се измества през входната функция f като използва представянето на транслационната група, за да се постигне транслация на всяка позиция. Тази операция по същество изпълнява съпоставяне на шаблони, при което k играе ролята на шаблон, който се сравнява с входа f във всички възможни транслации. Тъй като операцията включва линейна комбинация от транслирани ядра, тя е еквивариантна на транслацията, което означава, че всяка транслация на входа f води до съответна транслация на изхода:

$$f(T_{\mathbf{g}}(x)) = S_{\mathbf{g}}(f(x)) \tag{3.1}$$

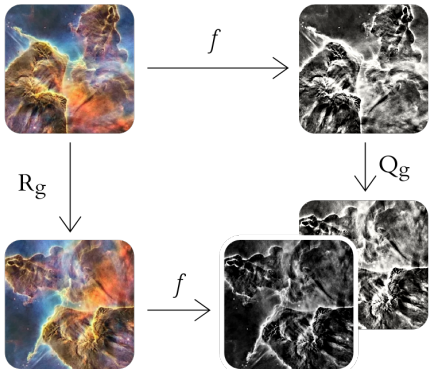
Това обаче не се отнася за операцията ротация, които не кому-

тира със стандартната конволюция.

$$f(R_{\mathbf{g}}(x)) \neq Q_{\mathbf{g}}(f(x)) \tag{3.2}$$



Фигура 3.1: Илюстрация на еквивариантността на стандартната операция конволюция при транслация на изходното изображение (у-ние (3.1).)



Фигура 3.2: Илюстрация на *отсъствието* на еквивариантност на стандартната операция конволюция при ротация на изходното изображение (у-ние (3.2).

3.1.2 Повдигаща корелация

За да може преобразуването между слоеве да е еквивариантно както по отношение на транслациите, така и на ротациите, въвеждаме модифициран корелационен оператор. Това се постига чрез дефиниране на повдигаща корелация (lifting correlation), която включва

както транслация, така и ротация. В тази разширена форма ядрото се завърта и премества според елементите на комбинираната група (рото-транслационна група). След това скаларното произведение се изчислява на всяка позиция, което позволява на оператора да съпостави модели във всички възможни транслации и ротации.

$$
\begin{aligned}
(k\tilde{\star}f)(\mathbf{x}) &= \left(\mathcal{L}_g^{SE(2)\to\mathbb{L}_2(\mathbb{R}^2)}k, f\right)_{\mathbb{L}_2(\mathbb{R}^2)} \\
&= \left(\mathcal{L}_{\mathrm{x}}^{\mathbb{R}^2\to\mathbb{L}_2(\mathbb{R}^2)}\mathcal{L}_\theta^{SO(2)\to\mathbb{L}_2(\mathbb{R}^2)}k, f\right)_{\mathbb{L}_2(\mathbb{R}^2)}
\end{aligned}
\tag{3.3}
$$

3.1.3 Конструиране на групови конволюционни мрежи

С концепцията за повдигащи корелации е възможно конструирането на групови конволюционни невронни мрежи (G-CNN). Тези мрежи се състоят от повдигащи слоеве, групови конволюционни слоеве и проекционни слоеве. Груповите конволюции оптимизират работата на модела, като вместо да прилагат трансформации върху целия входен сигнал (например ротации или отражения), те ги прилагат върху самите филтри. Това позволява на мрежата да обработва данните по-ефективно и да запази еквивариантността спрямо определени симетрии, без да увеличава значително изчислителната сложност. Чрез този процес се създава структуриран набор от признаци (накратко набор от признаци), съдържащ активации за всеки възможен еле-

мент от групата. [9]

Едновременното отчитане на ротационната и транслационната еквивариантности се постига чрез нов тип преобразувания, които са полупряко произведение на горните две. Така се получава групата SE(2) - специалната евклидова група.

Еквивариантните модели намаляват броя на необходимите параметри чрез споделяне на параметри (weight sharing), което води до по-ефективно използване на ресурсите и повторна употреба на параметрите при различни трансформации на входа. Този подход работи ефективно за дискретни групи от трансформации, като например ротации, където е дефиниран краен набор от различни трансформации. Въпреки това, когато се опитваме да разширим този подход към по-големи или дори безкрайни групи от трансформации, като непрекъснато въртене, възникват практически предизвикателства, които изискват допълнителни изследвания за разработване на техники, които могат ефективно да се справят със сложността на непрекъснатите трансформации.

3.2 Еквивариантни преобразувания спрямо групата на Евклид

Евклидовата група $E(2)$ представлява всички трансформации на равнината $\mathbb{R}^2$, които запазват разстоянията, включително трансла-

ции, ротации и отражения. Тези трансформации са важни, тъй като характерните модели в изображенията могат да се появят във всяка позиция и ориентация. Това е особено важно за изображения без фиксирана глобална ориентация, като сателитни изображения.

Евклидовата група $E(2)$ може да се разглежда като комбинация от две групи: групата на транслациите $(\mathbb{R}^2, +)$ и ортогоналната група $O(2)$, която включва всички операции, които поддържат началото фиксирано, като ротации и отражения.

Тази комбинация се извършва с помощта на математическа операция, наречена "полупряко произведение"(semidirect product):

$$E(2) \cong (\mathbb{R}^2, +) \rtimes O(2)$$

За да се справим с различни нива на симетрия в изображенията, ние също разглеждаме подгрупи на евклидовата група от вида $(\mathbb{R}^2, +) \rtimes G$, където G е подсгрупа на $O(2)$. Например, G може да бъде:

- Специалната ортогонална група $SO(2)$, която включва непрекъснати ротации без отражения.
- Групата $(\{\pm 1\}, *)$, представляваща отражения по определена ос.
- Циклични групи C_N, представляващи N дискретни ротации.

- Диедрални групи D_N, които включват N дискретни завъртания и отражения.
- Самата ортогонална група $O(2)$, която включва всички непрекъснати ротации и отражения.

3.3 Управляеми ядра

Управляемите ядра (steerable kernels) са ключова концепция при конструирането на конволюционни невронни мрежи (CNN), които са еквивариантни при по-широк диапазон от трансформации. Основната идея зад управляемите ядра е да се проектират филтри, които могат да бъдат "насочвани"или завъртани, за да съответстват на функции във входно изображение при различни трансформации, като ротации и отражения. Това прави управляемите ядра от решаващо значение за изграждането на еквивариантни невронни мрежи, като групови конволюционни невронни мрежи (G-CNN) и $E(2)$-еквивариантни конволюционни невронни мрежи (E2CNN), които трябва да се справят ефективно със сложни симетрии. [3] [11]

3.3.1 Определение

Казва се, че едно ядро $k : \mathbb{R}^d \to \mathbb{R}$ е управляемо, ако може да бъде изразено като линейна комбинация от краен набор от базисни филтри $\psi_i : \mathbb{R}^d \to \mathbb{R}$, които трансформират предсказуемо под действието

на трансформационна група G. За трансформация $g \in G$, ядрото k може да бъде написан като:

$$k(g \cdot x) = \sum_{i=1}^{n} \alpha_i(g)\psi_i(x),$$

където $\alpha_i(g)$ са коефициентите на управление, които зависят от трансформацията g [6]. Това уравнение показва, че всяка завъртяна или отразена версия на ядрото k може да бъде синтезирана чрез подходящо претегляне на базовите филтри ψ_i.

3.3.2 Еквивариантност чрез управляеми ядра

Управляемите ядра правят конволюционните невронни мрежи (CNNs) еквивариантни спрямо трансформационна група G, като гарантират, че операцията на конволюция е комутативна с действието на групата. По-конкретно, конволюционен слой, който използва управляеми ядра, е еквивариантен спрямо група G, ако прилагането на трансформация $g \in G$ към входа води до съответната трансформация на изхода. [4]Математически това може да се изрази по следния начин:

$$[\mathcal{T}_g f * k]\,(x) = \mathcal{T}_g\,[f * k]\,(x),$$

където $\mathcal{T}$ представлява груповото действие върху входната функция f, а $*$ означава операцията на конволюция [2]. Чрез проектирането на

ядрото k да бъде управляемо, се гарантира, че конволюцията зачита симетрията на групата G. Например, ако $G = E(2)$, групата на всички транслации, завъртания и отражения в равнината, управляемите ядра могат да бъдат конструирани, за да поддържат еквивариантност при тези преобразувания.

Глава 4

Модел на еквивариантна невронна мрежа за астрономически изображения

С установените теоретичните основи на теорията на групите, груповите действия и еквивариантните невронни мрежи, сега се обръщаме към практическото приложение на тези концепции за анализ на астрономически изображения. Тази глава описва внедряването на еквивариантни невронни мрежи (E2CNN) с помощта на пакета escnn в PyTorch [11][8], приложен към набора от данни Galaxy Zoo.

Анализът на астрономически изображения поставя много предизвикателства поради широкия диапазон от мащаби, ориентации и сложни морфологии на небесните обекти. Традиционните конволюционни невронни мрежи (CNN), макар и ефективни за много задачи по разпознаване на изображения, не отчитат по подразбиране симетриите като ротации и отражения, които често се срещат в астрономическите данни. За да преодолеем това ограничение, прилагаме E2CNN, които са специално проектирани да бъдат еквивариантни спрямо евклидовата група $E(2)$, което позволява на мрежата да разпознава характеристики независимо от тяхната ориентация или позиция в изображението. Този подход не само подобрява способността на модела да научава различни последователности в астрономическите изображения, но и повишава неговата способност за обобщаване, като използва симетриите, присъстващи в данните.

Следващите раздели ще предоставят подробно обяснение на използвания набор от данни, внедрената специфична E2CNN архитектура, експерименталната настройка и получените резултати. Започваме с общ преглед на набора от данни Galaxy Zoo, последван от ръководство стъпка по стъпка за настройка на E2CNN в PyTorch, включително дизайн на мрежовата архитектура, процедура за обучение и показатели за оценка.

4.1 Преглед на набора от данни

Наборът от данни Galaxy Zoo е широкомащабна колекция от астрономически изображения, създадени чрез гражданска научна инициатива, в която милиони доброволци участваха във визуално класифициране на галактики въз основа на тяхната морфология. Наборът от данни се състои от изображения, получени основно от Sloan Digital Sky Survey (SDSS) и включва различни видове галактики, като спирални, елиптични и неправилни. Класификациите включват допълнителни подробности като броя на спиралните ръкави, наличието на прегради и ориентацията на галактиката. Те предоставят богат набор от лейбъли, които улавят разнообразните структурни характеристики на галактиките. Изображенията варират по размер, разделителна способност и яркост. Това разнообразие прави набора от данни Galaxy Zoo ценен ресурс за обучение и оценка на модели за машинно обучение, особено в контекста на задачи, които изискват устойчивост на ротационни и други пространствени симетрии. В настоящата работа използваме подмножество от изображенията в Galaxy Zoo, които сме разделили в две категории - round и cigar. Всяка категория има по 260 изображения за обучение на модела и 40 за оценка на точността му. Те се намират в GitHub хранилището на разработката [5].

За да се подготвят данните за приложения за машинно обучение, изображенията преминават през няколко стъпки за предварителна

обработка, включително ръчно отсяване на некачествените изображения, нормализиране, преоразмеряване до последователно измерение и техники за увеличаване, като завъртане и обръщане, за да се увеличи устойчивостта на позиционни вариации. Тези стъпки на предварителна обработка гарантират, че наборът от данни е много подходящ за обучение на усъвършенствани модели като тези, използващи еквивариантни невронни мрежи, които се възползват от присъщите ротационни симетрии, присъстващи в астрономическите изображения.

4.2 Имплементация на E2CNN в PyTorch

4.2.1 Въведение в пакета escnn

Пакетът `escnn` е специализирана библиотека, предназначена за внедряване на еквивариантни конволюционни невронни мрежи (E2CNN) по отношение на непрекъснати симетрии, като евклидовата група $E(2)$, която включва транслации, ротации и отражения. Тази библиотека разширява традиционните конволюционни невронни мрежи (CNN), за да включи еквивариантни трансформации, позволявайки на мрежата да зачита по своята същност симетриите, налични в данните. Това е особено полезно за задачи, при които модели или обекти могат да се появят в произволни позиции и ориентации, каквито са астрономическите изображения.

Основните функции на пакета `escnn` включват:

- **Групова еквивариантност:** Предоставя инструменти за създаване на конволюционни слоеве, които са еквивариантни на трансформации от различни групи (напр. транслации, ротации).

- **Управляеми ядра:** Пакетът позволява внедряването на управляеми филтри, които са основна част от осигуряването на еквивариантност към непрекъснатите трансформации.

- **Гъвкави групови представяния:** escnn поддържа различни групови представяния, позволявайки на потребителите да дефинират еквивариантност по отношение на широк диапазон от групи, от прости ротации до по-сложни симетрични групи.

- **Интеграция с PyTorch:** Пакета има подобен на PyTorch интерфейс, което улеснява интегрирането в съществуващи процеси за дълбоко обучение (deep learning). Функции като групови конволюции и повдигащи слоеве могат да бъдат реализирани точно както биха се използвали стандартни конволюционни слоеве в PyTorch.

`escnn` е съдържа няколко подпакета:

- `escnn.group`, който включва основни понятия и операции с групи и представяния;

- `escnn.kernels`, чрез който дефинират управляеми ядра;
- `escnn.gspaces`, чрез който се дефинира евклидовото пространство и неговите симетрии;
- `escnn.nn`, който съдържа типове еквивариантни слоеве за дълбоки НМ.

4.2.2 Дизайн на мрежовата архитектура

Архитектурата E2CNN, използвана в тази задача, е проектирана да използва свойствата на симетрията на данните чрез използване на еквивариантни конволюционни слоеве. Това гарантира, че мрежата остава чувствителна към присъщите ротационни, транслационни и отражателни симетрии.

1. **Входен слой:** Входният слой получава предварително обработени изображения, които са преоразмерени, за да позволят оптимално използване на филтри с нечетен размер. Тази предварителна обработка включва допълване на изображенията до 29×29 и използване на трансформации за намаляване на артефактите, когато изображенията претърпят ротация. Входното изображение се третира като скаларно поле, което съответства на тривиалното представяне на групата $C8$ (групата от 45-градусови ротации). За да се справи с това, входният тензор е

обвит в геометричен тензор, който носи информация за свързаното с него групово представяне.

2. **Еквивариантни конволюционни слоеве:** Вместо традиционните конволюционни слоеве, тази архитектура използва еквивариантни конволюции. Всеки конволюционен слой в модела извършва групово-еквивариантна конволюция, използвайки филтри, които са проектирани да бъдат еквивариантни по отношение на ротационната група $C8$. Това гарантира, че картите на характеристиките (feature map), произведени от мрежата, се трансформират по подходящ начин при ротации.

   - 1-ви слой: Първият слой се състои от 24 карти на характеристиките, всяка от които се трансформира съгласно регулярното представяне на C8. Размерът на ядрото е 7×7, а след конволюцията се прилага комбинация от batch нормализация и ReLU активация.

     Кодът, с който се създава този слой (`block1`) е следния:

```
r2_act = gspaces.rot2dOnR2(N=8)
in_type = nn.FieldType(r2_act, r2_act.trivial_repr])
out_type = nn.FieldType(r2_act, 24*[r2_act.regular_repr])
block1 = nn.SequentialModule( nn.MaskModule(in_type, 29,
   margin=1),
  nn.R2Conv(in_type, out_type, kernel_size=7, padding=1,
   bias=False),
  nn.InnerBatchNorm(out_type),
  nn.ReLU(out_type, inplace=True)
```

```
)
```

Listing 4.1: Създаване на първи конволюционен слой

`FieldType` указва на пространството, в което се намира сигнала, в първия параметър, както и на действията, които се осъществяват върху него, явяващи се представяния на групата, като втори параметър. Съществената операция в 4.1 е обозначена с `R2Conv`. Тя трансформира 8те ротирани сигнала в 24 карти на характеристиките.

- 2-ри слой: Вторият слой увеличава броя на картите на характеристиките до 48, с ядро 5×5. Отново се прилага batch нормализация и ReLU активизация. След това точков слой за осреднено обединяване с антипаразитна обработка (Anti-aliasing) намалява пространствените измерения наполовина, като същевременно запазва свойството на мрежата да реагира по същия начин на трансформации като транслации.

  Кодът, с който се създава втория слой (`block2`) е сходен с този за първия слой 4.1 и има следния вид:

```
in_type = block1.out_type
out_type = nn.FieldType(r2_act, 48*[r2_act.regular_repr])
block2 = nn.SequentialModule(
  nn.R2Conv(in_type, out_type, kernel_size=5, padding=2,
   bias=False),
  nn.InnerBatchNorm(out_type),
```

```
6     nn.ReLU(out_type, inplace=True)
7 )
```

Listing 4.2: Създаване на втори конволюционен слой

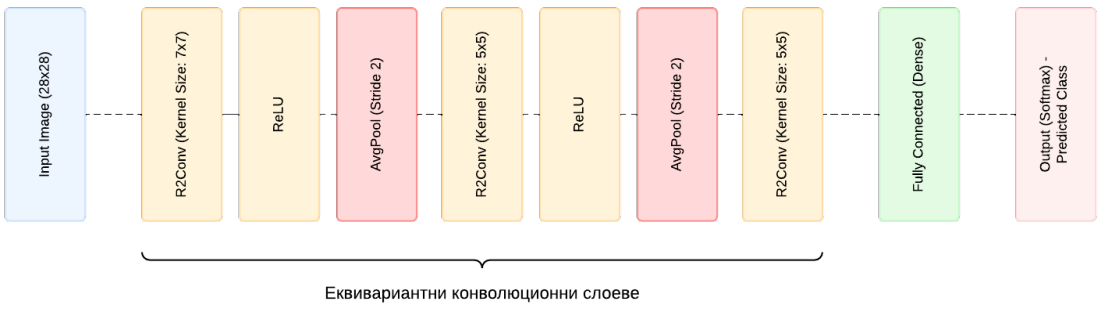
- 3-ти и 4-ти слой: Всеки от тези слоеве поддържа 48 карти на характеристиките, използвайки 5×5 ядра. И двата слоя са последвани от партидна нормализация и ReLU активиране. След четвъртия слой друг слой за осреднено обединяване допълнително намалява измеренията.
- 5-ти и 6-ти слой: Броят на картите на функции се увеличава до 96 в петия слой и остава на 96 в шестия. И двата слоя използват ядро 5×5, последвано от batch нормализация и ReLU активация. След шестия слой, последният антиалиасиран осреднен обединяващ слой намалява пространствените измерения, за да се подготви за глобално обединяване.

3. **Групово обединяване и напълно свързани слоеве:** След конволюционните слоеве, картите на характеристиките преминават през операция за групово обединяване, която намалява картите на характеристиките по груповото измерение, като запазва само най-информативните характеристики. Обединеният изход след това се подава към напълно свързан слой (fully-connected) за генериране на крайната класификация. Напълно

свързаната мрежа се състои от:

- Линеен слой с 64 изходни единици;
- Слой за нормализация на партидите (batch normalization);
- ELU функция за активация;
- Краен линеен слой за генериране на изходните логове за 5-те целеви класа.

Този подход, слой по слой гарантира, че мрежата може ефективно да осъществява трансформации в данните, без да губи критична информация за симетрията, което е от съществено значение за класификацията на астрономически изображения на галактики.



Фигура 4.1: Архитектура на модела.

Пълният код се намира в GitHub хранилището с кода и данните на разработката [5].

4.2.3 Обучение на модела

Процесът на обучение на модела включва няколко компонента, включително избор на функция за загуба, техники за оптимизация и показатели за оценка. Тук е описан всеки аспект от процедурата за обучение, използвана за обучение на модел C8 Steerable CNN.

1. **Предварителна обработка и увеличаване на данните**

   За да се обучи моделът ефективно, входните данни претърпяват поредица от трансформации, които помагат за разширяване на набора от данни и смекчаване на пренастройването:

   - Допълване: Всяко изображение се допълва до размер 29×29, за да се осигури по-добро подравняване по време на конволюцията с филтри с нечетен размер.
   - Преоразмеряване: За да се избегнат артефакти от интерполация при ротация, изображенията първо се увеличават с коефициент 3 (до размер 87×87), след което се завъртат и отново се преоразмеряват до 29×29.
   - Случайна ротация: По време на обучението изображенията преминават през случайни ротации до 180 градуса, което гарантира, че моделът ще научи ротационна инвариантност. Това е особено важно с оглед на естеството на задачата, при която мрежата трябва да се справя с произволни ротации на входните данни.

- Нормализация: Изображенията се преобразуват в тензори и се нормализират, за да се гарантира, че стойностите на всички пиксели са правилно мащабирани за използване в мрежата.

2. **Функция на загубата**

   Функцията на загубата, използвана в тази задача, е загуба на кросентропия, която е често срещан избор за проблеми с многокласова класификация. Тази функция на загуба сравнява предсказания изход на модела с истинските етикети и изчислява грешката. Формулата е следната:

$$\mathcal{L} = -\sum_{i=1}^{N} t_i \log(y_i)$$

   където t_i е истинският етикет, а y_i е предсказаната вероятност за клас i. Кросентропията налага "наказания"за предсказания, които се отклоняват от истинския етикет, и насърчава модела да генерира по-високи вероятности за правилния клас.

3. **Оптимизатор**

   Оптимизаторът, използван за актуализиране на параметрите на модела, е Adam. Adam е алгоритъм за оптимизация с адаптивна скорост на обучение, който регулира скоростта на обучение за всеки параметър поотделно, базирайки се на оценки

на първите и вторите моменти на градиентите. Конкретните параметри, използвани в случая, са:

- Коефициент на спускане: 5×10^{-5}, относително ниска стойност, избрана с цел осигуряване на надеждна сходимост.
- Намаляване на теглото (weight decay): 1×10^{-5}, което помага за предотвратяване на пренасищане (overfitting) чрез налагане на наказания за големи тегла, като ефективно служи като форма на $L2$ регуляризация.

Оптимизаторът Adam е подходящ за тази задача заради способността си да се справя с разредени градиенти и ефективната си изчислителна производителност.

4. **Цикъл на обучение**

Цикълът на обучение преминава през набора от данни за 31 епохи, изпълнявайки следните стъпки при всяка итерация:

- Прав ход (Forward Pass): Входните изображения се подават през модела, за да се генерират предсказания за всяка партида.
- Изчисляване на загубата (Loss Computation): Загубата на кросентропия се изчислява чрез сравняване на предсказанията с истинските етикети.

- Обратен ход (Backward Pass): Градиентите на загубата спрямо всеки от параметрите на модела се изчисляват чрез обратно разпространение (backpropagation).
- Стъпка на оптимизатора (Optimizer Step): Параметрите на модела се актуализират с помощта на изчислените градиенти и оптимизатора Adam.

За обучение използваме стандартен цикъл:

```
for epoch in range(NUM_EPOCHS):
  model.train()
  for i, (x, t) in enumerate(train_loader):
    optimizer.zero_grad()
    x = x.to(device)
    t = t.to(device)
    y = model(x)
    loss = loss_function(y, t)
    loss.backward()
    optimizer.step()
```

Listing 4.3: Цикъл на обучение

5. **Оценка и точност**

За да се следи производителността на модела, оценяването се извършва на редовни интервали (на всеки 10 епохи) върху тестовия набор от данни:

- Изчисляване на точността: Предсказанията от модела се сравняват с истинските етикети, а точността се изчисля-

ва като съотношение на правилно предсказаните етикети спрямо общия брой на пробите в тестовия набор.

```
with torch.no_grad():
  model.eval()
  for i, (x, t) in enumerate(test_loader):
    x = x.to(device)
    t = t.to(device)
    y = model(x)
    _, prediction = torch.max(y.data, 1)
    total += t.shape[0]
    correct += (prediction == t).sum().item()
    print(f"epoch {epoch} | test accuracy: {correct/total
   *100.}")
```

Listing 4.4: Код за оценка на точността на прогнозирането

Цикълът за оценка протича без изчисляване на градиенти, за да се намали употребата на памет и да се увеличи скоростта.

Характерни резултати от изпълненението на обучителния цикъп 4.3 с внедрен код за оценка на точността (вж 4.4) са показани по-долу:

```
epoch 0 | test accuracy: 20.833333333333336
epoch 10 | test accuracy: 85.83333333333333
epoch 20 | test accuracy: 94.16666666666667
epoch 30 | test accuracy: 95.83333333333334
epoch 40 | test accuracy: 97.5
epoch 50 | test accuracy: 96.66666666666667
```

```
epoch 60 | test accuracy: 98.33333333333333
epoch 70 | test accuracy: 98.33333333333333
epoch 80 | test accuracy: 98.33333333333333
```

6. **Регуляризация и предотвратяване на пренасищане**

   За намаляване на пренасищането се използват няколко стратегии:

   - Увеличаване на данните (Data Augmentation): Случайни ротации на тренировъчните данни помагат на модела да се генерализира по-добре при непознати ротации на входа, като осигуряват, че той научава ротационно-инвариантни характеристики.
   - Намаляване на теглото (Weight Decay): Използването на намаляване на теглото в оптимизатора Adam помага за предотвратяване на пренасищане, като възпрепятства модела да научава прекалено сложни модели на данните от обучението, действайки ефективно като $L2$ регуляризация.
   - Нормализация на партидите (Batch Normalization): След всеки конволюционен слой се прилага нормализация на партидите, за да се нормализира изходът, което подобрява стабилността по време на обучението и служи като фор-

ма на регуляризация, като намалява чувствителността на модела към вариации в входните данни.

Тази комбинация от техники осигурява, че моделът не само научава да се адаптира към тренировъчните данни, но също така се обобщава добре към непознати тестови данни.

Глава 5

Заключение

Разбработката в настоящата дипломна работа показва, че еквивариантни преобразования по отношение на ротации, приложени към два класа астрономически изображения, дава много добри резултати в решаването на задачата за класификация на подобни изображения. Въпреки добрите резултати, успехът може да се смята само за частичен поради следните непълноти в изследването:

- количеството на изображенията за обучение на модела са относително малко на брой;
- не е направено сравнение с *по-конвенционален* модел с конволюционна невронна мрежа;
- не е експериментирано с по-разнообразни преобразувания, както и с вариране дълбочината на мрежата.

Посочените пропуски са в процес на отстраняване и по темата се оформя публикация в научно издание.

Библиография

[1] Yoshua Bengio, Yann Lecun, and Geoffrey Hinton. Deep learning for ai. *Commun. ACM*, 64(7):58–65, June 2021.

[2] Taco S. Cohen, Mario Geiger, Jonas Koehler, and Max Welling. Spherical cnns, 2018.

[3] Taco S. Cohen and Max Welling. Group equivariant convolutional networks, 2016.

[4] Taco S. Cohen and Max Welling. Steerable cnns, 2016.

[5] Yoana Foteva. E2CNN-AstroAnalysis . `https://github.com/yoanaFoteva/E2CNN-AstroAnalysis`, 2024. [Online; accessed 19-Oct-2024].

[6] William T Freeman, Edward H Adelson, et al. The design and use of steerable filters. *IEEE Transactions on Pattern analysis and machine intelligence*, 13(9):891–906, 1991.

[7] James E Humphreys. *Introduction to Lie algebras and*

representation theory, volume 9. Springer Science & Business Media, 2012.

[8] Erik Jenner and Maurice Weiler. Steerable partial differential operators for equivariant neural networks. In *International Conference on Learning Representations*, 2022.

[9] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. the handbook of brain theory and neural networks. *MIT Press*, 3361(10):1995, 1995.

[10] David W. Romero, Erik J. Bekkers, Jakub M. Tomczak, and Mark Hoogendoorn. Attentive group equivariant convolutional networks, 2020.

[11] Maurice Weiler and Gabriele Cesa. General E(2)-Equivariant Steerable CNNs. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.