https://doi.org/10.31891/2219-9365-2025-81-24
УДК 004.05

БЕДРАТЮК Ганна
Хмельницький національний університет
https://orcid.org/0000-0003-0224-5549
e-mail: bedratyuk@ukr.net

ЗГОРТКОВА НЕЙРОННА МЕРЕЖА З ПРОЕКТИВНО-ІНВАРІАНТНИМ ПУЛІНГОМ

В статті розглядається проблема класифікації зображень, до яких застосовані проективні перетворення, та пропонується архітектура згорткової нейронної мережі (CNN), що включає проективно-інваріантний пулінговий шар. На відміну від класичних афінних перетворень, для яких існують відомі еквіваріантні перетворення (контрольовані згорткові нейронні мережі, гармонійні H-Nets тощо), задача знаходження проективної еквіваріантності залишається відкритою. В статті зроблено крок у напрямку розв'язання цієї проблеми і запропоновано реалізацію проективно-інваріантного пулінгу. Порівняно із звичайною CNN, ми демонструємо, що додавання такого пулінгу покращує робастність нашої мережі до проективних викривлень. Експерименти проводяться на наборах зображень proMNIST і rotoMNIST, згенерованих із стандартного набору MNIST відповідними перетвореннями.

Ключові слова: Згорткові нейронні мережі, Проективні перетворення, Інваріантний пулінг, Робастність, Аугментації зображень

BEDRATYUK Anna
Khmelnytskyi National University

CONVOLUTIONAL NEURAL NETWORK WITH PROJECTIVE INVARIANT POOLING

This paper addresses the challenging problem of image classification under projective transformations and presents a novel approach by incorporating a projective invariant pooling layer into a convolutional neural network (CNN) architecture. While classical affine transformations have been extensively studied, with well-established equivariant architectures such as steerable convolutional neural networks and harmonic H-Nets, achieving projective equivariance remains an open problem in the field of deep learning.

To bridge this gap, we introduce a method that extends CNNs by integrating a specialized pooling layer designed to be invariant to projective distortions. This enhancement allows the network to maintain performance and robustness when faced with significant geometric transformations that would otherwise degrade classification accuracy. Our proposed pooling mechanism ensures that the feature extraction process remains stable despite changes in perspective, making it particularly useful for applications involving images captured from varying viewpoints.

To validate our approach, we conduct extensive experiments on the proMNIST and rotoMNIST datasets, which we generate by applying projective and rotational transformations to the standard MNIST dataset. Through comparative analysis with conventional CNN architectures, we demonstrate that our method significantly improves classification robustness under projective distortions. The results highlight the potential of incorporating projective invariance into deep learning models, paving the way for further advancements in geometric deep learning and practical applications in fields such as remote sensing, medical imaging, and autonomous navigation.

Keywords: Convolutional Neural Networks, Projective Transformations, Invariant Pooling, Robustness, Image Augmentations

ПОСТАНОВКА ПРОБЛЕМИ У ЗАГАЛЬНОМУ ВИГЛЯДІ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

У багатьох реальних задачах (розпізнавання тексту, аналіз знімків із різних ракурсів, обробка документів) зустрічаються зображення, які зазнають проективних перетворень, що деформують їх суттєвіше за прості повороти та зсуви (наприклад, паралельні лінії сходяться в точку на горизонті). Хоча трансляційна еквіваріантність згорткових нейронних мереж (CNN) вже стала стандартом, а методи типу H-Nets і Steerable CNN успішно розв'язують задачу еквіваріантності до поворотів, для проективної групи загального методу все ще не існує через її суттєву складність. У межах Геометричного глибокого навчання постає питання про створення універсальної мережі, еквіваріантної до будь-яких геометричних груп, проте з практичної точки зору наразі зручніше вбудовувати в нейронну мережу окремі інваріанти. Наша робота робить перший крок до часткової проективної інваріантності через спеціальний пулінг-шар, який дозволяє знизити чутливість CNN до проективних спотворень, не змінюючи суттєво архітектуру нейронної мережі.В статті запропоновано проективно-інваріантний пулінг який реалізується через обчислення дискретного інтегралу який є інваріантом відносно проективного перетворення. Для перевірки ефективності застосування проективно-інваріантний пулінгу пропонується три архітектури CNN: Базова класична CNN як орієнтир, CNN з ппроективно-інваріантний у фіналі і CNN з у якій проективно-інваріантний пуліег винесено в окрему паралельну гілку. Експерименти проводилися на згенерованих набір зображень proMNIST, rotoMNIST, які дозволяють оцінити, наскільки мережа лишається ефективною, коли зображення «перекошені» через

випадкові проективні матриці чи великі кути повороту.

Результати показують, що нейронні мережі з такою архітектурою дійсно підвищує точність класифікації наборів даних на яких діє проективна група, наближаючи нас до *повноцінної* (хоч і не абсолютної) проективної еквіваріантності.

АНАЛІЗ ДОСЛІДЖЕНЬ ТА ПУБЛІКАЦІЙ

Як показують численні дослідження, запит на геометрично узгоджені згорткові архітектури підтверджує свою актуальність, зокрема й у свіжих роботах. Ранні дослідження в цьому напрямку базуються на класичних ідеях щодо поворотної еквіваріантності: Steerable CNN [13, 17, 4] забезпечують певну інваріантність до груп перетворень SO(2) (повороти) та D8 (дискретні віддзеркалення). У [14] було запропоновано гармонійні мережі (H-Nets), що використовують гармонійну згортку для досягнення ефективної обертової еквіваріантності в евклідовому просторі. Теоретичні аспекти побудови групово-еквіваріантних згорток узагальнено у [2, 7, 6], де обговорюються калібрувально-теоретичні підходи. Подальші дослідження [11, 16, 15] розглядають можливості створення універсальних інваріантних/еквіваріантних архітектур без втрати точності класифікації. Зокрема, [11] підкреслює, що в певних умовах такі мережі здатні апроксимувати широкий клас функцій при коректно вбудованій інваріантності. У роботі [15] розглядається шкала-інваріантність через Riesz-бази, що є додатковим прикладом розширення ідей контрольованих CNN. Хоча значна частина літератури [13, 16, 18, 14] фокусується на поворотах і паралельних перенесеннях, недавні праці [5, 9] вказують на відкрите питання щодо проективної групи. У [5] автори прямо зазначають складність розробки еквіваріантних підходів для проективних та афінних перетворень. Аналогічно, недавні публікації [9, 10, 12] показують, що навіть у випадку загальних груп Лі залишається багато невирішених теоретичних проблем, особливо коли йдеться про реальні багатоканальні зображення з високою роздільною здатністю.

Причина складності класифікації дії проективної групи полягає у тому, що вона має 8 параметрів, у той час як, наприклад, група поворотів однопараметрична, тому побудувати контрольовані еквіваріантні фільтри для неї виявилося набагато складніше. Отже методи, що *наближають* інваріантність до проективних викривлень, фактично знаходяться у стадії активного дослідження. Наша ідея з проективно-інваріантним пулінгом — одне з можливих рішень, яке не претендує на абсолютну еквіваріантність, але демонструє покращені результати на реальних даних.

ПРОЕКТИВНИЙ ІНВАРІАНТ ТА ЙОГО ОБЧИСЛЕННЯ

Дійсна проективна лінійна група $PGL(2,\mathbb{R})$ є восьмипараметричною групою Лі, яка утворена проективними лінійними перетвореннями площини

$$T\colon (x,y) \to \left(\frac{a_1x + a_2y + a_3}{c_1x + c_2y + c_3}, \frac{b_1x + b_2y + b_3}{c_1x + c_2y + c_3}\right), (x,y) \in \mathbb{R}^2,$$

з якобіаном

$$J(T) = \frac{D}{(xc_1 + yc_2 + c_3)^3}, D = \begin{vmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_2 \end{vmatrix}.$$

Напівтонове зображення ототожнимо з тричі диференційовною на $\mathbb{R}^2$ функцією $u = f(x,y)$, яка відмінна від нуля в деякій обмеженій області площини. Частинні похідні за змінними x, y будемо позначати таким чином: $u_x, u_y, u_{xx}, \ldots$.

Нагадаємо, що диференціальним інваріантом ваги k для групи G називається функція $F = F(x, y, u, u_x, u_y, \ldots)$, яка залежить від змінних, функції u та її частинних похідних, така що для всіх елементів $T \in G$ групи виконується тотожність

$$T \cdot F = J(T)^k F, \forall T \in G.$$

Диференціальний інваріант ваги $k = 0$ називається абсолютним інваріантом проективної групи G. Абсолютні інваріанти цікаві тим що вони не змінюються при проективних перетвореннях, тобто для них виконується така тотожність:

$T \cdot F = F, \forall T \in G.$

Знаходження диференціальних інваріантів є складною обчислювальною задачею, оскільки продовження дії групи на похідні є досить громіздким і ми не будемо їх наводити тут. Ми скористаємося результатами роботи [3] в якій явно обчислено деякі проективні інваріанти. Зокрема в цій роботі доведено, що наступні два диференціальні вирази:

$$R_0 = u_y^2\, u_{xx} \; - \; 2\, u_y\, u_x\, u_{xy} \; + \; u_{yy}\, u_x^2,$$

та

$$R_1 = u_x^2(u_{xxy}u_{yyy} - u_{xyy}^2) + u_x u_y(u_{xxy}u_{xyy} - u_{xxx}u_{yyy}) + u_y^2(u_{xxx}u_{xyy} - u_{xxy}^2)$$
$$+2u_x(u_{yy}^2 u_{xxx} - 3u_{xy}u_{yy}u_{xxy} + (u_{xx}u_{yy} + 2u_{xy}^2)u_{xyy} - u_{xx}u_{xy}u_{yyy})$$
$$+2u_y(-u_{xy}u_{yy}u_{xxx} + (u_{xx}u_{yy} + 2u_{xy}^2)u_{xxy} - 3u_{xx}u_{xy}u_{xyy} + u_{xx}^2 u_{yyy}) - 4(u_{xx}u_{yy} - u_{xy}^2)^2,$$

є диференціальними інваріантами ваг -2 і 4 відповідно.

Тому вираз

$$I = \frac{R_1}{R_0^2}$$

буде абсолютним диференціальним інваріантом, який ми і використаємо при проектуванні нашої нейронної мережі.

На практиці, зображення $f(x,y)$ є дискретною прямокутною таблицею розміру $H \times W$. Тому для обчисленні дискретних похідних ми використовуємо метод центральних різниць. Хоча традиційно використовуються 3-точкові схеми, що відповідають ядрам 3×3), для підвищення точності ми застосовуємо 5-точкові схеми, які утворюють ядра розміром 5×5 на двовимірному зображенні.

Для обчислення перших похідних використовується наступний одновимірну згортку:

$$\frac{1}{12}[-1,\ 8,\ 0,\ -8,\ 1]$$

Тоді, для створення двовимірного ядра 5×5, яке відповідає оператору дифренціювання по x, ці коефіцієнти розміщуються в центральному рядку матриці, а всі інші елементи заповнюються нулями:

$$\frac{1}{12}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ -1 & 8 & 0 & -8 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Аналогічно, для частинної похідної по y, використовується той самий шаблон, але розташований у центральному стовпці матриці:

$$\frac{1}{12}\begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 8 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -8 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Для других похідних використовується одновимірна згортка:

$$\frac{1}{12}[1,\ -16,\ 30,\ -16,\ 1]$$

Тоді для для u_{xx} отримуємо таку двовимірну згортку

$$\frac{1}{12}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & -16 & 30 & -16 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

а для u_{yy}:

$$\frac{1}{12}\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -16 & 0 & 0 \\ 0 & 0 & 30 & 0 & 0 \\ 0 & 0 & -16 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Для обчислення змішаної похідної u_{xy}, щоб отримати матрицю фільтра потрібно вектор коефіцієнтів $\frac{1}{12}[-1,8,0,-8,1]$ як вектор-стовпець та цей же вектор транспонований як вектор-рядок. Їх добуток дає 5×5-матрицю:

$$\frac{1}{12}\begin{bmatrix} -1 \\ 8 \\ 0 \\ -8 \\ 1 \end{bmatrix} \cdot \frac{1}{12}[-1,8,0,-8,1] = \frac{1}{144}\begin{bmatrix} 1 & -8 & 0 & 8 & -1 \\ -8 & 64 & 0 & -64 & 8 \\ 0 & 0 & 0 & 0 & 0 \\ 8 & -64 & 0 & 64 & -8 \\ -1 & 8 & 0 & -8 & 1 \end{bmatrix}$$

Аналогічно ми будуємо фільтри для обчислення похідних вищих порядків, які зустрічаються у виразах для диференціальних інваріантів. Ці ядра застосовується для обчислення змішаної похідної через операцію згортки.

Практична реалізація згортки може бути виконана кількома способами. У середовищі OpenCV

використовується функція cv2.filter2D з параметром ddepth=CV_32F та відповідним 5×5 ядром, що забезпечує згортку з автоматичним відображенням країв. В середовищі TensorFlow можна застосувати функцію tf.nn.depthwise_conv2d, яка дозволяє використати однакове ядро 5×5 до всіх каналів зображення за допомогою depthwise згортки.

При реалізації важливо враховувати кілька технічних аспектів. Для правильного вирівнювання використовують anchor=(2,2) для ядра 5×5 або параметр padding='same' у TensorFlow. У середовищі TensorFlow ядро потребує перетворення розмірності до $(5,5,1,1)$ з наступним тиражуванням (tile) відповідно до кількості каналів зображення.

Обробка країв зображення є важливим аспектом реалізації, оскільки при застосуванні ядра 5×5 потрібен відступ у 2 пікселі від кожного краю ($\lfloor 5/2 \rfloor = 2$). Це може призвести до спотворення значень похідних через відсутність необхідних сусідніх пікселів. Для вирішення цієї проблеми можна використовувати режими REPLICATE або SYMMETRIC, які заповнюють відсутні значення копіюванням або віддзеркаленням крайових пікселів. Альтернативним підходом є свідоме ігнорування крайової смуги шириною 2 пікселі для мінімізації похибок обчислення.

Особливу увагу слід приділяти обробці шуму, який суттєво впливає на точність обчислення других та змішаних похідних. Значення похідних вищих порядків можуть демонструвати значні коливання, або набувати екстремальних значень на зашумлених ділянках. Ефективним методом боротьби з цією проблемою є попередня фільтрація зображення гаусовим 3×3-фільтром. Це значно знижує чутливість похідних другого порядку до дрібномасштабного шуму.

Використання згорток з ядрами розміром 5×5 для обчислення похідних різних порядків (включаючи змішані) забезпечує більш точні результати порівняно з простішими 3×3 схемами типу оператора Собеля. Хоча такий підхід вимагає більших обчислювальних ресурсів, він надає необхідну точність для подальшого формування інтегрального проективно-інваріантного функціонала.

АРХІТЕКТУРИ НЕЙРОННОЇ МЕРЕЖІ ТА НАБОРИ ДАНИХ

Ми розглянемо три варіанти згорткових нейронних мереж (CNN) для задачі класифікації зображень, на яких діють проективні перетворення. Перша модель є базовою, без спеціальних інваріантів, і слугує еталоном для порівняння ефективності двох інших мереж. Друга мережа містить проективно-інваріантний пулінг у кінці, а третя реалізує паралельну схему з окремою гілкою для проективного пулінгу. Ми розглядаємо невеликі CNN, оскільки нашою метою є лише ілюстрація того, що введений нами теоретичний проективний інваріант справді реагує на проективні перетворення, а не отримання найкращої точності класифікації за рахунок збільшення глибини та ширини мережі.

Базова (звичайна) CNN є типовою послідовністю шарів Conv2D, Pooling та Dense. Структура включає кілька згорткових шарів (Conv2D) для вилучення ознак, де кожен шар може йти після нелінійності (ReLU) та можливого пулінгу. Далі йде MaxPooling2D (або AveragePooling2D) для зменшення розміру активацій зі збереженням суттєвих ознак. Завершується структура шарами Flatten та Dense для класифікації на 10 вихідних класів (у випадку MNIST). Ця базова мережа слугує еталоном порівняння - ми очікуємо, що вона покаже високу точність на недефомованих даних, але гірше працюватиме на сильно проективно деформованих зображеннях. Структурований звіт про архітектуру цієї нейронної мережі наведено нижче:

```
Layer (type)                 Output Shape              Param #
=================================================================
 conv2d_65 (Conv2D)          (None, 26, 26, 32)        320

 max_pooling2d_54 (MaxPoolin  (None, 13, 13, 32)       0
 g2D)

 dropout_48 (Dropout)        (None, 13, 13, 32)        0

 conv2d_66 (Conv2D)          (None, 11, 11, 64)        18496

 max_pooling2d_55 (MaxPoolin  (None, 5, 5, 64)         0
 g2D)

 dropout_49 (Dropout)        (None, 5, 5, 64)          0

 flatten_30 (Flatten)        (None, 1600)              0

 dense_65 (Dense)            (None, 128)               204928

 dropout_50 (Dropout)        (None, 128)               0

 dense_66 (Dense)            (None, 10)                1290

=================================================================
Total params: 225,034
Trainable params: 225,034
Non-trainable params: 0
```

CNN із проективним пулінгом у кінці зберігає основні елементи класичної CNN, але замінює фінальний глобальний пулінг на ProjectiveInvariantPooling_5x5. Мережа починається з кількох блоків Conv2D + Pool, які зменшують просторові розміри зі збереженням суттєвих ознак. У фіналі замість стандартного пулінгу використовується кастомний шар, який обчислює дискретні похідні $dx, dy, d2x, d2y, dxy$, формує інваріантну складову та обчислює проективний інваріант I по всій просторовій сітці. Після пулінгу отримуємо вектор розміру (batch, channels), який проходить через Dense шари з ReLU та softmax активаціями. Такий підхід дозволяє мережі спочатку вивчити просторові ознаки через згортки, а потім глобально інтегрувати проективну інформацію. Структурований звіт про архітектуру цієї нейронної мережі наведено нижче:

Layer (type)	Output Shape	Param #
conv2d_67 (Conv2D)	(None, 26, 26, 32)	320
max_pooling2d_56 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_68 (Conv2D)	(None, 11, 11, 64)	18496
projective_invariant_pooling_5 (ProjectiveInvariantPooling)	(None, 64)	0
flatten_31 (Flatten)	(None, 64)	0
dense_67 (Dense)	(None, 128)	8320
dense_68 (Dense)	(None, 10)	1290

Total params: 28,426
Trainable params: 28,426
Non-trainable params: 0

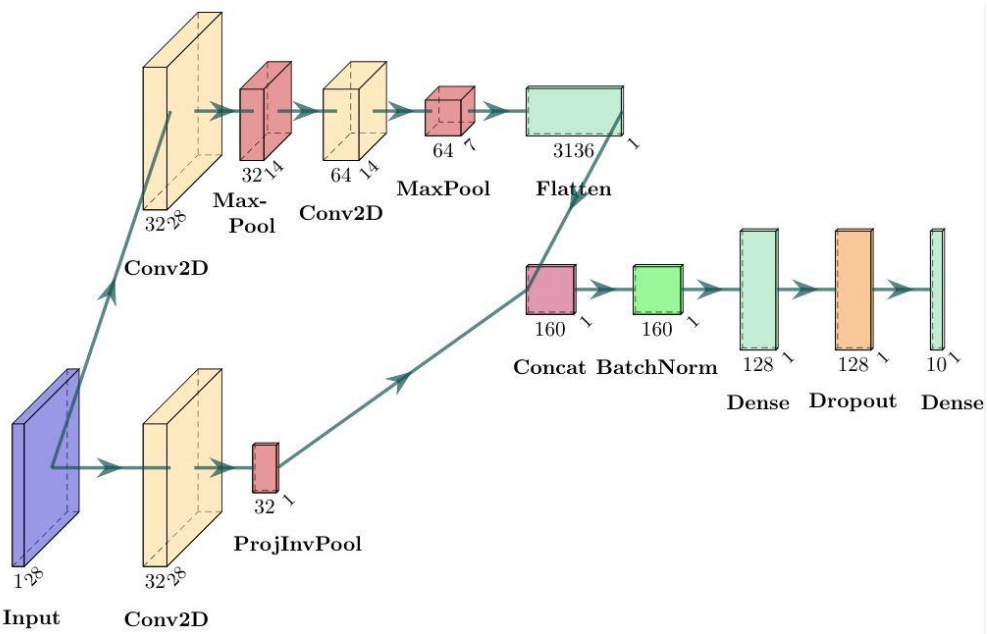

Рис 1. Згорткова нейронна мережа з паралельними гілками та проективно-інваріантним пулінгом (зображення створене засобами PlotNeuralNet)

Паралельна архітектура, Ми розгалужуємо мережу на дві гілки. Перша гілка виконує звичайну обробку (Conv + Pool + Flatten + Dense), а друга застосовує проективно-інваріантний пулінг безпосередньо до вхідного зображення після початкової згортки Conv(32). На виході обидві гілки об'єднуються через Concatenate і передаються у спільні Dense шари. Така архітектура дозволяє поєднати класичний метод вилучення ознак, який добре працює на стандартних зображеннях, з додатковою інформацією, що робить модель стійкою до сильніших проективних викривлень. Структурований звіт про архітектуру цієї нейронної мережі наведено нижче:

```
Layer (type)                 Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 28, 28, 1)]       0

 conv2d_10 (Conv2D)          (None, 28, 28, 32)        320

 max_pooling2d_6 (MaxPooling2D)  (None, 14, 14, 32)    0

 conv2d_11 (Conv2D)          (None, 14, 14, 64)        18496

 max_pooling2d_7 (MaxPooling2D)  (None, 7, 7, 64)      0

 conv2d_12 (Conv2D)          (None, 28, 28, 32)        320

 flatten_3 (Flatten)         (None, 3136)              0

 projective_invariant_pooling_5  (None, 32)            0
 x5_4 (ProjectiveInvariantPooli
 ng_5x5)

 dense_7 (Dense)             (None, 128)               401536

 batch_normalization_2 (BatchNo  (None, 32)            128
 rmalization)

 concatenate_1 (Concatenate)     (None, 160)           0


 batch_normalization_3 (BatchNo  (None, 160)           640
 rmalization)

 activation_1 (Activation)   (None, 160)               0

 dense_8 (Dense)             (None, 128)               20608

 dropout_2 (Dropout)         (None, 128)               0

 dense_9 (Dense)             (None, 10)                1290

=================================================================
Total params: 443,338
Trainable params: 442,954
Non-trainable params: 384
```

Таким чином, ми маємо три архітектури, які будуть порівнюватися на штучно створених наборах даних proMNIST та rotoMNIST: базову CNN як еталон, CNN з інваріантним пулінгом у кінці як просте рішення, та паралельну CNN з проективним пулінгом як комплексний підхід для узгодження звичайних та інваріантних ознак.

Для проведення експериментів ми використали дві модифікації стандартного набору даних MNIST. Звичайний MNIST містить 70 000 розмічених зображеннь розміру (28×28) із рукописними цифрами, які досить добре розпізнаються класичними CNN. Однак для імітації «проективних» викривлень ми виконуємо наступні кроки:

1. Спочатку до кожного зображення (28×28) додається рамка з нулів в 5 пікселів. Нулі колір на краях зменшує ймовірність того, що проективні викривлення виходитиме за межі зображення.
2. Далі для кожного такого розширеного зображення генерується випадкова матриця проективного перетворення (3×3). Ми обираємо 4 кути зображення і зсуваємо їх на невеликі випадкові значення, а потім виконуємо cv2.warpPerspective. В результаті цифра виявляється «перекошеною» чи порушеною перспективою, що імітує певну проективну трансформацію.

Утворений набір (proMNIST) має значно більшу різноманітність геометричних викривлень, ніж звичайний MNIST і він підходить для перевірки того, чи здатна мережа з проективно-інваріантним пулінгом краще узагальнювати і зберігати точність на таких «скривлених» цифрах.

Інший набір даних rotoMNIST фокусується не на загальних проективних перетвореннях, а на поворотах та масштабуваннях. Кожне зображення повертається на випадковий кут $\theta \in [90°..270°]$ а потім

масштабується з випадковим коефіцєнтом з діапазону$[0.5..1.1]$. На відміну від proMNIST, де маємо різні проективні трансформації, у rotoMNIST єдиним, але досить сильним викривленням є поворот на діапазон кутів, які не спостерігалися в оригінальному MNIST. Такий набір даних часто використовується, щоб показати, як сильно падає точність базових CNN, якщо мережа не бачила (під час тренування) подібних рандомних «крутильних» змін.

Таким чином, proMNIST і rotoMNIST є двома ключовими датасетами, які демонструють, як звичайна мережа починає «збоїти» (порівняно із базовими 98–99% на звичайному MNIST), натомість мережа з проективно-інваріантним пулінгом може зберегти суттєво кращу точність на «скривлених» прикладах. Обидві вибірки є похідними від оригінального MNIST, тому кількість зображень, розмітка (мітки {0..9}) лишаються незмінними.

ЕКСПЕРИМЕНТИ ТА РЕЗУЛЬТАТИ

У реалізації моделі використовувався високорівневий API Keras на мові програмування Python, що забезпечило зручність та швидкість розробки. Всі шари нейронної мережі були визначені за допомогою послідовного або функціонального підходів Keras, що дозволило гнучко налаштовувати архітектуру моделі. Особливу увагу приділено інтеграції спеціалізованого шару проективно-інваріантної згортки, який було реалізовано як кастомний клас у Keras, забезпечуючи необхідну функціональність для підвищення стійкості моделі до геометричних трансформацій. Для оптимізації процесу навчання використовувалися різноманітні колбеки Keras, що допомогло уникнути перенавчання та покращити загальну продуктивність моделі. Крім того, була здійснена передобробка даних із використанням бібліотек NumPy та Pandas для ефективного управління та аугментації вхідних зображень. Завдяки використанню Keras, процес налагодження гіперпараметрів та експериментування з різними архітектурними рішеннями став максимально ефективним, що сприяло досягненню високих результатів у задачах класифікації.

Опишемо експерименти з порівняння ефективності трьох архітектур нейронних мереж: базової CNN, CNN із проективно-інваріантним пулінгом у фіналі та паралельної мережі з окремою гілкою проективного пулінгу. Оцінювалися точність класифікації (accuracy), F1-метрика та перевірялася стійкість до значних трансформацій зображень. Основним показником порівняння є точність класифікації на тестовій вибірці. F1-метрика розглядається додатково у випадках, коли важливий баланс Precision/Recall, хоча в проведених експериментах вона корелює з accuracy. Особлива увага приділяється стійкості моделей до сильних трансформацій - для цього використовуються згенеровані набори proMNIST, rotoMNIST.

Результати експериментів наведені в Таблиці 1.

Таблиця 1.

Порівняння результатів на тестових наборах proMNIST і rotoMNIST

Модель	proMNIST		rotoMNIST	
	Accuracy	F1	Accuracy	F1
Звичайна CNN	0.965	0.963	0.930	0.928
CNN + проектив. пулінг	0.972	0.970	0.941	0.939
Паралельна мережа	0.975	0.973	0.946	0.944

Експерименти показали суттєву перевагу архітектур з проективно-інваріантним пулінгом. На наборі proMNIST базова CNN досягає точності 96.5%, тоді як додавання інваріантного пулінгу підвищує цей показник до 97.2%. Паралельна архітектура демонструє найкращий результат - 97.5%. Аналогічна тенденція спостерігається на наборі rotoMNIST, де різниця між моделями стає ще помітнішою. Важливо відзначити, що F1-метрика майже повністю корелює з accuracy для всіх моделей, що свідчить про збалансованість класифікації між різними класами цифр. Це особливо важливо, оскільки вказує на відсутність систематичних помилок при розпізнаванні певних цифр після трансформацій.

Аналіз ефективності інваріантного пулінгу. При тестуванні на звичайному MNIST різниця між моделями виявляється незначною, що очікувано, враховуючи відсутність складних трансформацій у цьому наборі. Однак при збільшенні параметрів деформації переваги інваріантного пулінгу стають очевидними:

При максимальному зсуві зображень (max_shift):

Базова CNN: падіння точності до 90 CNN з пулінгом: стабільні 93 Паралельна архітектура: збереження точності на рівні 94

При комбінованих трансформаціях (масштабування + поворот):

Базова CNN демонструє найбільше падіння точності Моделі з інваріантним пулінгом зберігають високу стабільність Паралельна архітектура показує найкращу стійкість до комбінованих викривлень

Аналіз помилок класифікації виявив, що:

Базова CNN найчастіше помиляється на цифрах з сильним нахилом Моделі з інваріантним пулінгом краще справляються з різними типами деформацій Паралельна архітектура найефективніше обробляє екстремальні випадки викривлень

Проведені експерименти переконливо демонструють, що проективно-інваріантний пулінг значно підвищує стійкість нейронних мереж до геометричних трансформацій вхідних даних. Особливо важливо, що цей ефект посилюється при збільшенні складності трансформацій, що робить дану технологію перспективною для реальних застосувань, де вхідні дані можуть мати значні геометричні викривлення.

ВИСНОВКИ З ДАНОГО ДОСЛІДЖЕННЯ І ПЕРСПЕКТИВИ ПОДАЛЬШИХ РОЗВІДОК У ДАНОМУ НАПРЯМІ

У роботі запропоновано та досліджено метод проективно-інваріантного пулінгу для покращення стійкості згорткових нейронних мереж до геометричних трансформацій зображень. Експерименти показали, що додавання такого пулінгу, особливо в паралельній архітектурі, дозволяє підвищити точність класифікації на 1-3% при сильних проективних викривленнях порівняно з базовою CNN. Найбільший ефект спостерігається на наборах даних proMNIST та rotoMNIST, де присутні значні геометричні деформації. Хоча запропонований метод має певні обмеження, пов'язані з обчислювальною складністю та чутливістю до шуму, він демонструє перспективний підхід до побудови CNN з вбудованою геометричною інваріантністю. Подальші дослідження будуть спрямовані на оптимізацію обчислень, підвищення стабільності при сильних викривленнях та розширення методу на інші типи трансформацій зображень.

Література

1. Mumuni, A., & Mumuni, F. (2021). CNN Architectures for Geometric Transformation-Invariant Feature Representation in Computer Vision: A Review. SN Computer Science, 2, 340.

2. Gerken, J. E., Aronsson, J., Carlsson, O., et al. (2023). Geometric deep learning and equivariant neural networks. Artificial Intelligence Review, 56, 14605–14662.

3. Olver, P. J. (2023). Projective invariants of images. European Journal of Applied Mathematics, 34(5), 936-946.

4. Weiler, M., Forré, P., Verlinde, E., & Welling, M. (2023). Equivariant and Coordinate Independent Convolutional Networks - A Gauge Field Theory of Neural Networks.

5. Reisert, M. (2007). Equivariant holomorphic filters-theory and applications (Technical Report 3). Albert-Ludwig University Freiburg.

6. Bronstein, M. M., Bruna, J., Cohen, T., & Veličković, P. (2021). Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges. arXiv preprint arXiv:2104.13478.

7. Li, Y., Qiu, Y., Chen, Y., He, L., & Lin, Z. (2024). Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 5546-5556.

8. Shumaylov, Z., Zaika, P., Rowbottom, J., Sherry, F., Weber, M., & Schönlieb, C. B. (2024). Lie Algebra Canonicalization: Equivariant Neural Operators under arbitrary Lie Groups. arXiv preprint arXiv:2410.02698.

9. Samudre, A., Petrache, M., Nord, B. D., & Trivedi, S. (2024). Symmetry-Based Structured Matrices for Efficient Approximately Equivariant Networks. arXiv preprint arXiv:2409.11772.

10. Maron, H., Fetaya, E., Segol, N., & Lipman, Y. (2019). On the universality of invariant networks. In International Conference on Machine Learning (pp. 4363-4371). PMLR.

11. Mironenco, M., & Forré, P. (2023). Lie group decompositions for equivariant neural networks. arXiv preprint arXiv:2310.11366.

12. Cohen, T. S., & Welling, M. (2017). Steerable CNNs. International Conference on Learning Representations (ICLR).

13. Worrall, D. E., Garbin, S. J., Turmukhambetov, D., & Brostow, G. J. (2017). Harmonic networks: Deep translation and rotation equivariance. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 5028-5037).

14. Barisin, T., Schladitz, K., & Redenbach, C. (2024). Riesz Networks: Scale-Invariant Neural Networks in a Single Forward Pass. Journal of Mathematical Imaging and Vision, 66, 246–270.

15. Marcos, D., Volpi, M., & Tuia, D. (2016). Learning rotation invariant convolutional filters for texture classification. 2016 23rd International Conference on Pattern Recognition (ICPR). https://doi.org/10.1109/icpr.2016.7899932

16. Ravanbakhsh, S. (2020). Universal equivariant multilayer perceptrons. International Conference on Machine Learning (ICML).

References

1. Mumuni, A., & Mumuni, F. (2021). CNN Architectures for Geometric Transformation-Invariant Feature Representation in Computer Vision: A Review. SN Computer Science, 2, 340.

2. Gerken, J. E., Aronsson, J., Carlsson, O., et al. (2023). Geometric deep learning and equivariant neural networks. Artificial Intelligence Review, 56, 14605–14662.

3. Olver, P. J. (2023). Projective invariants of images. European Journal of Applied Mathematics, 34(5), 936-946.

4. Weiler, M., Forré, P., Verlinde, E., & Welling, M. (2023). Equivariant and Coordinate Independent Convolutional Networks - A Gauge Field Theory of Neural Networks.

5. Reisert, M. (2007). Equivariant holomorphic filters-theory and applications (Technical Report 3). Albert-Ludwig University Freiburg.

6. Bronstein, M. M., Bruna, J., Cohen, T., & Veličković, P. (2021). Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges. arXiv preprint arXiv:2104.13478.

7. Li, Y., Qiu, Y., Chen, Y., He, L., & Lin, Z. (2024). Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 5546-5556.

8. Shumaylov, Z., Zaika, P., Rowbottom, J., Sherry, F., Weber, M., & Schönlieb, C. B. (2024). Lie Algebra Canonicalization: Equivariant Neural Operators under arbitrary Lie Groups. arXiv preprint arXiv:2410.02698.

9. Samudre, A., Petrache, M., Nord, B. D., & Trivedi, S. (2024). Symmetry-Based Structured Matrices for Efficient Approximately Equivariant Networks. arXiv preprint arXiv:2409.11772.

10. Maron, H., Fetaya, E., Segol, N., & Lipman, Y. (2019). On the universality of invariant networks. In International Conference on Machine Learning (pp. 4363-4371). PMLR.

11. Mironenco, M., & Forré, P. (2023). Lie group decompositions for equivariant neural networks. arXiv preprint arXiv:2310.11366.

12. Cohen, T. S., & Welling, M. (2017). Steerable CNNs. International Conference on Learning Representations (ICLR).

13. Worrall, D. E., Garbin, S. J., Turmukhambetov, D., & Brostow, G. J. (2017). Harmonic networks: Deep translation and rotation equivariance. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 5028-5037).

14. Barisin, T., Schladitz, K., & Redenbach, C. (2024). Riesz Networks: Scale-Invariant Neural Networks in a Single Forward Pass. Journal of Mathematical Imaging and Vision, 66, 246–270.

15. Marcos, D., Volpi, M., & Tuia, D. (2016). Learning rotation invariant convolutional filters for texture classification. 2016 23rd International Conference on Pattern Recognition (ICPR). https://doi.org/10.1109/icpr.2016.7899932

16. Ravanbakhsh, S. (2020). Universal equivariant multilayer perceptrons. International Conference on Machine Learning (ICML).