**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**

**Радіотехнічний факультет**

**Кафедра прикладної радіоелектроніки**

«На правах рукопису»
УДК _____

До захисту допущено:
В.о. зав. кафедрою
_____ Михайло Степанов
«___»_____2021 р.

# Магістерська дисертація

**на здобуття ступеня магістра**

**за освітньо-професійною програмою «Радіозв'язок і оброблення сигналів»**

**за спеціальністю 172 «Телекомунікації та радіотехніка» на тему:**
**«Діагностика електронного модулю з використанням штучної нейронної мережі»**

Виконав
студент 2 курсу, групи РА-01мп
Ткаченко Руслан Юрійовчи ___ _____

Керівник:
Ст. викл.
Адаменко Володимир Олексійович

Рецензент:
Д.т.н., проф.
Калюжний Олександр Якович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент ___ _____

Київ – 2021 року

**Національний технічний університет України**
**«Київський політехнічний інститут імені Ігоря Сікорського»**
**Радіотехнічний факультет**
**Кафедра прикладної радіоелектроніки**

Рівень вищої освіти – другий (магістерський)

Спеціальність – 172 Телекомунікації та радіотехніка

Освітньо-професійна програма «Радіозв'язок і оброблення сигналів»

ЗАТВЕРДЖУЮ
В.о.завідувача кафедри
_____ Михайло СТЕПАНОВ
«___» _____ 2021 р.

# ЗАВДАННЯ
# на магістерську дисертацію студента
# Ткаченко Руслан Юрійович

1.Тема дисертації «Діагностика електронного модулю з використанням штучної нейронної мережі»
науковий керівник дисертації Адаменко Володимир Олексійович, старший викладач
затверджені наказом по університету від «15» листопада 2021 р. №3744-с

2. Термін подання студентом дисертації 12 грудня 2021 року

3. Об'єкт дослідження: електронні компоненти встановлені на друкованій платі

4. Вихідні дані: зображення компонентів електронного модулю_____

5. Перелік завдань, які потрібно розробити: огляд дефектів компонентів електронного модуля, вибір програмного забезпечення для реалізації нейронних мереж, синтезувати нейронну мережу, провести навчання та тестування нейронної мережі.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: презентація

7. Орієнтовний перелік публікацій _____

_____

9. Дата видачі завдання 02 вересня 2021 року

Календарний план

| № з/п | Назва етапів виконання магістерської дисертації | Термін виконання етапів магістерської дисертації | Примітка |
|---|---|---|---|
| 1 | Огляд дефектів компонентів електронного модуля | 02.09.2021-13.09.2021 | |
| | Огляд програмного забеспечення для реалізації нейронних мереж | 14.09.2021-31.09.2021 | |
| | Дослідження можливостей DeepLearning toolbox | 01.10.2021-13.10.2021 | |
| | Огляд класів та типів архітектур популярних нейронних меререж | 14.10.2021-30.10.2021 | |
| | Синтезування нейронної мережі | 1.11.2021-14.11.2021 | |
| | Тестування нейронної мережі | 15.11.2021-13.12.2021 | |
| | | | |
| | | | |

Студент Руслан ТКАЧЕНКО

Науковий керівник Володимир АДАМЕНКО

# РЕФЕРАТ

Магістерська дисертація містить в собі 42 сторінки, 2 таблиці, 16 зображень, 1 додаток і 8 бібліографічних найменувань.

Актуальність теми: підвищення продуктивності виробництва шляхом збільшення швидкості проходження візуального контролю електронних модулів для виявлення дефектів є актуальною задачею.

Метою магістерської роботи є розроблення та тестування нейронної мережі для пошуку дефектів електронних модулів.

Наукова новизна: науковою новизною є штучна нейронна мережа яка дає змогу швидко класифікувати компоненти на електронних модулях.

У результаті отримано згорткову нейронну мережу яка розпізнає основні компоненти електронних модулів, а саме діоди, конденсатори, транзистори, резистори та мікросхеми з достовірністю в 96%.

**Ключові слова**: нейронна мережа, дефект, електронний модуль.

## ABSTRACT

The master's thesis includes 42 pages, 16 images, 2 tables, 1 appendix and 11 bibliographic references.

The aim of the master's thesis is to create a neural network and learn to diagnose an electronic module by comparing it with a reference image.

Relevance of the topic: there is a study of convolutional neural network Squeezenet on a variety of components of the electronic module and search for defects in them.

Scientific novelty: The scientific novelty is the artificial neural network, which allows you to quickly recognize the components on the electronic modules and search for defects in them.

As a result, a convolutional neural network that recognizes the main components of electronic modules, namely diodes, capacitors, transistors, resistors and microcircuits and checks for defects is obtained.



**Keywords**: neural network, defect, electronic module.

# ПОЯСНЮВАЛЬНА ЗАПИСКА
## до магістерської дисертації

на тему: Діагностика електронного модулю з використанням штучної нейронної мережі

Київ — 2021 рік

**ЗМІСТ**

# ПЕРЕЛІК СКОРОЧЕНЬ

ДП — Друкована плата
НМ — Нейронна мережа
ШНМ — Штучна нейронна мережа
ІМС — Інтегрована мікросхема
SMT — Surface Mount Technology
RNN — Recurrent neural network
LSTM — Long short-term memory
ESN — Echo State Networks
DNN — Деконволюційні нейронні мережі
ВСТ — Внутрішньосхемне тестування
NiN — Network-in-network

# ВСТУП

Сучасні тенденції виробництва передбачають усунення людини зі всіх ланок виробничого процесу. Проте є процеси, для виконання яких людина все ще залишається потрібною. До таких виробничих процесів можна віднести візуальний контроль. Одним з варіантів заміни людини на таких важливих ділянках виробництва є використання комп'ютерного зору.

Метою магістерської дисертації є застосування згорткових нейронних мереж для пошуку дефектів на електронних модулях, які виникають в процесі їх виробництва. До таких дефектів, перш за все, можна віднести не правильне встановлення компонентів на плату або їх зміщення в процесі проходження конвеєром. Тому штучна нейронна мережа повинна виокремлювати електронні компоненти на друкованих платах та мати змогу визначити тип компоненту та, власне, чи правильно він встановлений.

Наведені в роботі напрацювання можуть бути використані для подальшого удосконалення алгоритмів комп'ютерного зору в задачах візуального контролю електронних модулів на різних виробничих етапах.

# 1 ДЕФЕКТИ ЕЛЕКТРОННИХ КОМПОНЕНТІВ ТА МОДУЛІВ

Під час виробництва електронних модулів виникає багато факторів які впливають на виникнення дефектів. Для розроблення алгоритму їх виявлення необхідно провести огляд найпоширеніших типів дефектів для технології поверхневого монтажу (SMT — Surface Mount Technology).

Елементи ланцюга — це ті компоненти або електронна схема, яка може подавати енергію в ланцюг, або може приймати енергію з контуру. Всі компоненти ланцюга з'єднані між собою в потрібному порядку провідниками для формування бажаного і повного кола. Якщо з'єднувальний провідник вважається ідеальним, то всі частини ланцюга, за винятком з'єднувальних провідників, є елементами схеми.

Технологія поверхневого монтажу не є процесом пайки без дефектів. У даній магістерській роботі будуть розглянуті прості дефекти в технології поверхневого монтажу, через які друкована плата виходить з ладу. Зменшення таких дефектів є ключем до надання ефективних послуг зі складання друкованих плат.

Технологія поверхневого монтажу — це процес, у якому електричні компоненти монтуються безпосередньо на поверхню друкованої плати. Електричний компонент, змонтований таким чином, називається пристроєм поверхневого монтажу (SMD).

SMT має ряд переваг перед монтажем через отвір. SMT забезпечує легкість автоматизації та більш високу щільність компонентів на платі. Він також забезпечує більш високі швидкості ланцюга, забезпечуючи кращу високочастотну продуктивність [1].

Існує кілька відмінностей між технологією поверхневого монтажу та технологією монтажу в наскрізних отворах. Основні відмінності стосуються вартості, автоматизації, місця на платі та щільності компонентів. Наскрізні компоненти мають вищі витрати на виробництво, ніж компоненти SMT, і не підходять для автоматизації. Технологія наскрізних отворів також має вищі обмеження щодо простору на платі та щільності компонентів у порівнянні з SMT [1].

## 1.1 Види дефектів

**Помилка SMT № 1 — це коротке замикання паяним або електричним мостом.**

Міст для припою — це припій між двома провідниками, які не повинні бути електрично з’єднані, і які спричиняють електричне замикання. Електричні містки утворюються через надлишок припою, ці замикання призводять до несправності ланцюга. На рис 1.1 показано помилку SMT № 1.


Рисунок 1.1 — зображення КЗ електричний мостом.

Різноманітні причини можуть бути причиною такого замикання. Однак найбільш загальновизнаною причиною є проблема в процесі нанесення паяльної пасти. Розташування трафаретів до конфігурації друкованої плати може бути дещо неправильним. Надмірне відкладення паяльної пасти також може призвести до утворення містків. Це може статися, коли пропорція трафарету та прокладки занадто велика. Холодна паяльна паста може також викликати замикання [1].

Якщо область попереднього нагріву має надмірно повільну швидкість наростання, це може бути причиною замикання. Частина контакту з паяльною пастою може призвести до перекосу відкладення, що призведе до перемикання паяної пасти. Тривале замочування введе більше тепла до пасти і призведе до явища гарячого опадання пасти.

Неточність розміщення може ще більше зменшити проміжок між колодками. Отже, підвищуючи ймовірність перемикання. Занадто великий тиск на розміщення компонентів може видавити пасту з колодок.

Можливим рішенням є необхідність приділяти належну увагу вирівнюванню отворів трафарету та планшетів, якщо не використовується автоматичне вирівнювання принтера. А також забезпечити точковий тиск і точність розміщення компонентів.

**Помилка SMT № 2 Недостатня кількість паяних з’єднань або електричні розриви.**

Коли дві електрично з’єднані точки роз’єднуються, або область на друкованій платі, яка перериває передбачувану конструкцію ланцюга, називається електричним розмиканням.

Потенційними причинами виникнення даного дефекту це недостатня кількість припою на з’єднанні призведе до розриву ланцюга, якщо паяльна паста забивається в отвори трафарету. Навіть якщо об’єм припою достатній, може відбутися розмикання, якщо він не контактує як з проводом, так і з колодкою під час плавлення. Це називається компланарністю відведення компонентів. Розкриття також може бути наслідком самого процесу виготовлення друкованої плати.

Можливим рішенням перш за все є корекція співвідношення сторін. Співвідношення сторін визначається як відношення ширини діафрагми до товщини трафарету. Паяльна паста, що забиває отвори, може бути пов’язана з занадто малим співвідношенням сторін.

Екстремальні умови навколишнього середовища є суворим запереченням у виробничому процесі. Потрібно уникати забруднення паяльною пастою, контролюючі навколишнє середовище. Дослідження щодо компланарності також дуже важливо, коли справа доходить до вирішення електричних розривів.

**Помилка SMT № 3 — згортання припою.**

Розвиток дуже крихітних сферичних частинок припою, що ізолюють від основного тіла, що утворює з'єднання. Це є суттєвою проблемою для процесу без очищення, оскільки величезна кількість кульок припою може створити фальшивий міст між двома сусідніми провідниками, що спричиняє функціональні проблеми в електричному ланцюзі. Згортання припою не викликає такого великого занепокоєння з водорозчинними процедурами, оскільки вони регулярно видаляються під час процедури очищення [1].

На Рис.1.2 зображено дефект кульки припою.

Потенційними причинами даного дефекту є забруднення вологи паяльної пасти. Під час оплавлення волога насичується, залишаючи за собою сфери припою. Відсутність належного профілю оплавлення також може призвести до кульок припою. Швидка швидкість попереднього нагрівання не забезпечить достатньо часу для поступового випаровування розчинника. Надмірні оксиди на порошку припою в пасті також можуть утворювати кульки припою. Згортання припою може бути викликано поганим вирівнюванням друку паяльною пастою, а також коли паяльна паста надрукована на масці для припою замість прокладки.

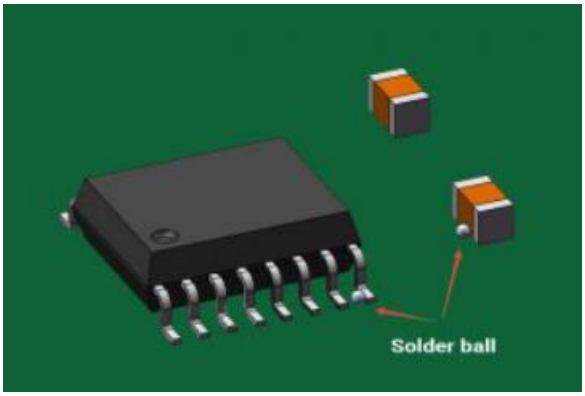


Рисунок 1.2 — Дефект кульки припою

Методи уникнення даного дефекту та рекомендації:

1. Рекомендується використовувати більш крупний порошок, оскільки дрібний порошок містить більше оксидів і має тенденцію швидше осипатися.
2. Процес оплавлення слід обирати відповідно до пайки.
3. Слід уникати взаємодії паяльної пасти з вологою та вологістю.
4. Перевірте використовуваний мінімальний тиск друку.
5. Вирівнювання друку слід перевіряти на послідовній основі, перш ніж приступати до перекомпонування.
6. Забезпечення правильного та частого очищення нижньої частини трафарету.

**Помилка SMT № 4 — ефект Манхеттена.**

Надгробна плита, яку іноді називають ефектом Манхеттена, — це компонент мікросхеми, який частково або повністю витягнув площадку у вертикальне положення, припаяний лише один кінець. На рис .1.3 наведено приклад дефекту Tombstoneing. Це є результатом дисбалансу сил під час процесу пайки оплавленням.

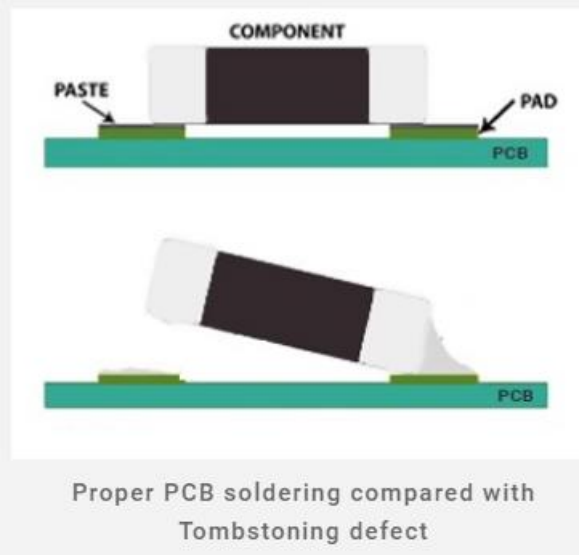


Рисунок 1.3 — Вигляд дефекту «надгробка» в SMT

Дефект «надгробка», потенційні причини:

1. Нерівномірний нагрів може спричинити перепади на клемах компонентів. Точніше, якщо розподіл тепла нерівномірний, то припій буде плавитися з різною швидкістю. Таким чином, одна сторона повертається перед іншою, що призводить до того, що інша встає прямо.
2. Нерівні радіатори, тобто заземлення, якщо вони присутні всередині шарів друкованої плати, можуть відводити тепло від майданчика.
3. Іноді через вплив температури та вологості на паяльну пасту недостатня сила паяльної пасти для утримання компонента на місці під час плавлення.
4. Надмірне рух під час та після операції оплавлення може спричинити зміщення компонентів, що призведе до захоронення.
5. Неоднакове розміщення компонентів на колодках перед оплавленням призводить до не збалансованих сил припою.

Можливі рішення:

- Корпус компонента повинен покривати щонайменше 50% обох колодок, щоб уникнути дисбалансу сил припою.
- Забезпечте високу точність розміщення компонентів.
- Рекомендується підтримувати високу температуру попереднього нагрівання, щоб під час оплавлення різниця між двома кінцями була невеликою.
- Зведення до мінімуму переміщення під час складання SMT до найменшого можливого під час плавлення.
- Найменший вплив екстремальних факторів навколишнього середовища, таких як висока температура і вологість.
- Розширена зона замочування може допомогти збалансувати силу змочування на обох накладках до того, як паста досягає стану розплавленого.

**Помилка SMT № 5 — Зневоднення.**

Стан у припаяному з'єднанні, при якому рідинний припій не чітко зчеплений принаймні з одним із компонентів. Стан, коли поверхня контактувала з рідким припоєм, проте, частина припою або жодна з них не тримається міцно.

Потенційні причини:

1. Однією з основних причин може бути погана обробка друкованої плати. Припустимо, що основний метал видно, як правило, його важче припаяти, отже, не відбувається змочування.
2. Це також може бути пов’язано з тим, що в процесі оплавлення занадто довгий час замочування. В результаті вичерпується флюс перед пайкою.
3. Можливо, під час процесу плавлення тепла недостатньо, отже, потік не досягає належної температури активації.

Можливі рішення:

- Необхідно адаптувати поверхню металу кращої якості, наприклад, термостійкий OSP або ENIG.
- Зменшення загального часу профілювання перед етапом оплавлення.
- Відповідний флюс для даної задачі пайки.
- Також прочитайте, як обійти чорну прокладку в обробці ENIG

**Помилка SMT № 6 — Пайка бісеру**

Утворення більших куль припою, розташованих поблизу дискретних компонентів, які мають дуже малу відстань один від одного. Ця деформація схожа на згортання припою, але вона дискретна в тому, як ці кульки припою міцно тримаються на дискретних компонентах, а не на гаджетах з кількома виходами [4].

Потенційні причини:

- Як правило, така проблема виникає через надмірну кількість паяної пасти, що відкладається.
- Іноді під час стадії попереднього розігріву відбувається відокремлення потоку, яке може подолати силу коалесценції пасти.
- Надмірний тиск на розміщення компонентів також може бути проблемою. Це може натиснути нанесену паяльну пасту на маску для пайки. Отже, неможливо злитися назад у суглоб.

Можливі рішення данного дефекту — це зменшення товщини трафарету або зменшення розмірів отвору. На тій стороні, де є валик припою, зменшення на 10% повинно вирішити цю проблему.

**Помилка SMT № 7 — Недостатнє заповнення та недостатня кількість припою.**

Кількість паяльної пасти, що наноситься на станцію принтера, набагато менша, ніж у конструкції отвору трафарету, або після оплавлення недостатньо припою, щоб утворити згин на виводах компонентів.

Отвір трафарету іноді може забиватися засохлою пастою. Це одна з основних причин проблеми. Під час циклу друку велике значення має достатній тиск по всій довжині леза ракеля. Занадто сильний тиск може спричинити зачерпування пасти, і це може призвести до виникнення данного дефекту. Паста не скочується в отвір через занадто високу швидкість роботи ракеля. Швидкість переміщення ракеля

визначає час, протягом якого паяльна паста скочується в отвори трафарету та на колодки друкованої плати.

Можливі рішення:

1. Великий отвір можна розділити на менші отвори і перевірити, чи немає занадто сильного тиску ракеля.
2. Трафарет необхідно очищати через регулярні проміжки часу, а пасту слід перевіряти на термін придатності чи сухості. Також слід забезпечити достатню підтримку дошки.
3. Небажана також надмірна швидкість роботи ракеля, яку також слід контролювати.

**Помилка SMT № 8 — Холодне з'єднання або зернисте з'єднання.**

Деякі паяні з'єднання іноді погано змочуються і мають сірий пористий вигляд після пайки. Його впізнають за темними, не відбиваючими, шорсткими поверхнями сплаву, які повинні бути яскравими і блискучими.

Однією з перерахованих основних причин є недостатня кількість тепла, що поглинається припоєм. Це відбувається через те, що присутнє тепло для оплавлення припою недостатньо. Занадто багато домішок у розчині припою також може призвести до дефекту.

Можливі рішення данного дефекту — це максимальна температура плавлення повинна бути достатньо високою, щоб матеріал був ретельно оплавлений. Збірка не повинна відчувати будь-яких рухів під час або відразу після оплавлення. А також необхідно провести аналіз сплаву, щоб перевірити наявність забруднень.

## 1.2 Методи пошуку дефектів

Одним із методів пошуку дефектів є внутрішньосхемне тестування (ВСТ). Внутрішньосхемне тестування — це потужний інструмент для тестування друкованих плат. За допомогою обладнання для внутрішньосхемне тестування можна отримати доступ до вузлів схеми на платі та виміряти продуктивність компонентів незалежно від інших компонентів, підключених до них. Вимірюються такі параметри як опір, ємність і тощо, а також робота аналогових компонентів, таких як операційні підсилювачі. Деякі функціональні можливості цифрових схем можуть бути виміряні, хоча їх складність зазвичай робить повну перевірку нерентабельною. Таким чином, використовуючи ВСТ, внутрішньосхемне тестування можна провести дуже повну перевірку друкованої плати, гарантуючи, що схема була виготовлена правильно і має дуже високі шанси працювати відповідно до своїх технічних характеристик.

Обладнання для внутрішньосхемне тестування забезпечує корисну та ефективну форму тестування друкованої плати, вимірюючи кожен компонент по черзі, щоб переконатися, що він на місці та має правильне значення. Оскільки більшість несправностей на платі виникають у процесі виробництва і зазвичай складається з коротких замикань, обривів або неправильних компонентів, така форма тестування дозволяє виявити більшість проблем на платі. Їх можна легко перевірити за

допомогою простих вимірювань опору, ємності та індуктивності інколи між двома точками на платі.

Також для визначення дефектів електронних модулів використовують метод периферійного сканування.

Периферійне сканування — вид структурного тестування друкованої плати з встановленими на неї компонентами, заснований на застосуванні деяких мікросхем стандарту IEEE 1149.1. Широко використовується також термін "граничне сканування". Результатом периферійного сканування є інформація про наявність в електроланцюгах типових несправностей, що виникають під час виробництва друкованих плат:

- коротких замикань,
- непропайок,
- западань на 0 чи 1,
- обривів доріжок,

Периферійне сканування було названо через те, що відповідні мікросхеми можуть за певних умов самі протестувати своє оточення — периферію на наявність несправностей [4].

**Візуальний контроль**

Для виявлення дефектів на електронних модулях використовують візуальні методи контролю, які полягають в огляді електронного модулю на наявність видимих дефектів, таких як перекос встановленого компоненту, містків з припою тощо.

Для проведення такого контролю використовують напівавтоматичні системи, коли комп’ютер порівнює частини зображення з еталонним та у разі виявлення розбіжностей виводить відповідну частину плати оператору, який вже приймає рішення чи є на зображення дефект чи не має.

Такі системи мають порівняно високий показник хибного виявлення проблеми, що не дозволяє позбавитися від оператора.

Провівши огляд видів дефектів електронного модулю та способів їх виявлення пропонується для оптимізації процесу візуального контролю використати штучні нейронні мережі (ШНМ), які успішно використовуються для оброблення зображення різної складності. Для використання ШНМ необхідно визначитися з програмним забезпеченням та архітектурою мережі.

# 2 ВИБІР ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РЕАЛІЗАЦІЇ НЕЙРОННОЇ МЕРЕЖІ

Нейронна мережа — математична модель, а також її програмне або апаратне втілення, побудована за принципом організації та функціонування біологічних нейронних мереж — мереж нервових клітин живого організму. Це поняття виникло щодо процесів, які протікають у мозку, і за спробі змоделювати ці процеси. Після розробки алгоритмів навчання одержувані моделі стали використовувати в практичних цілях: завдання прогнозування, розпізнавання образів, управління тощо [2].

## 2.1 Використання Python для реалізації ШНМ

Однією з мов програмування, яка часто використовується для вирішення прикладних задач, є мова Python. Ця мова має кілька переваг. Саме на базі цієї мови реалізована велика кількість бібліотек, які надають у зручному виді більшість доступних алгоритмів. Більшість із цих бібліотек використовують бібліотеку NumPy. Це бібліотека, яка дозволяє швидко і ефективно працювати з числовими даними, матрицями, таблицями чисел в різних форматах, проводити велику кількість типових операцій, що потрібні в процесі вирішення прикладних задач. В даній бібліотеці реалізовані десятки алгоритмів для задач кластеризації, регресії, класифікації, методу опорних векторів, лінійної та логістичної регресія та десятків інших. В кожному із доступних алгоритмів є велика кількість параметрів, які можна налаштувати під свою задачу.

Бібліотека Pandas є однією із дуже зручних бібліотек інтегрованої в Python, яка допомагає працювати з великою кількістю табличних даних. Вона дозволяє дуже швидко завантажувати дані, препроцесити їх (готувати у відповідний формат), щоб у зручному вигляді відправляти на опрацювання нашим алгоритмом [5].

На даний час завдяки популярності НМ на мові Python існує велика кількість бібліотек, які на досить різних рівнях абстракції (від низькорівневих до високо абстрагованих методів описання архітектури) надають можливість конструювання різноманітних НМ. Одна із досить поширених бібліотек низькорівневих операцій для реалізації алгоритмів НМ — бібліотека Theano. Вона реалізує комплексні матричні мультиплікації, швидкі методи згортки з множенням, вичленовуванням, методи регресії і всю backend-логіку роботи нейронних мереж. Одним із ключових бонусів таких бібліотек є те, що крім CPU реалізації (тобто реалізації роботи алгоритму на процесорі), бібліотеки типу Theano або TensorFlow (від Google) є open source, тобто з відкритим кодом (можна його подивитися чи дописати модулі, яких не вистачає). Це тільки дві з найпопулярніших бібліотек з реалізації backend-логіки НМ і це «конструктор», з якого можна зібрати НМ будь-якої архітектури або складності, модифікувати або скомбінувати методи навчання чи оптимізації. Один із ключових бонусів — в них реалізована підтримка обчислень на відеокартах. Процес роботи або тренування НМ пришвидшується у 70-300 разів у порівнянні із швидкістю роботи на процесорі, оскільки він виконує операції виконує послідовно,

по черзі. При виконанні обчислень на відеокарті, що являє собою систему з великої кількості маленьких процесорів, одна складна задача розбивається на велику кількість маленьких задач і вони виконуються паралельно. І саме бібліотеки Theano, TensorFlow досить ефективно використовують можливості GPU-обчислень (обчислення на відеокартах). Слід зауважити, що між відеокартою, мовою Python та бібліотекою, написаною на цій мові є ще один прошарок – CUDA – набір бібліотек, реалізований компанією NVidia, які дозволяють ефективно і швидко проводити обчислення на її відеокартах [5].

## 2.2 Використання C++ для реалізації ШНМ

C++ — компільована, статично типізована мова програмування загального призначення [6].

Підтримує такі парадигми програмування, як процедурне програмування, об'єктно-орієнтоване програмування, узагальнене програмування. Мова має багату стандартну бібліотеку, яка включає поширені контейнери і алгоритми, введення-виведення, регулярні висловлювання, підтримку багатопоточності та інші можливості. C++ поєднує властивості як високорівневих, і низькорівневих мов. У порівнянні з його попередником — мовою C – найбільшу увагу приділено підтримці об'єктно-орієнтованого та узагальненого програмування [6].

C++ широко використовується для розробки програмного забезпечення, будучи однією з найпопулярніших мов програмування. Область його застосування включає створення операційних систем, різноманітних прикладних програм, драйверів пристроїв, додатків для систем, високопродуктивних серверів, а також ігор. Існує безліч реалізацій мови C++ як безкоштовних, так і комерційних і для різних платформ. Наприклад, на платформі x86 це GCC, Visual C++, Intel C++ Compiler, Embarcadero (Borland) C++ Builder та інші. C++ вплинув інші мови програмування, насамперед Java і C#.

Для реалізації ШНМ на мові програмування C++ використовується одна із найпопулярніших бібліотек OpenNN (Open Neural Networks Library). Це програмна бібліотека, написана мовою програмування C++, яка реалізує нейронні мережі, основний напрямок досліджень у галузі глибокого навчання. Бібліотека має відкритий вихідний код та ліцензована під GNU Lesser General Public License. Ця глибока архітектура дозволяє проектувати нейронні мережі з універсальними властивостями апроксимації. Крім того, вона дозволяє програмувати багатопроцесорну обробку за допомогою OpenMP, щоб збільшити продуктивність комп'ютера.

OpenNN містить алгоритми машинного навчання як набору функцій. Вони можуть бути вбудовані в інші програмні інструменти за допомогою прикладного програмного інтерфейсу для інтеграції завдань прогнозної аналітики. У цьому плані графічний інтерфейс користувача відсутня, але деякі функції можуть бути підтримані спеціальними інструментами візуалізації [7].

## 2.3 Використання Matlab для реалізації ШНМ

Для реалізації ШНМ також підходить програмне забезпечення MathLab а саме додаток Deep Learning toolbox. Перевагою якого є простота, зручність та швидкість використання а також відсутність поглиблених знань програмування [8].

Deep Learning Toolbox надає основу для проектування та впровадження глибоких нейронних мереж з алгоритмами, попередньо підготовленими моделями та додатками. В даному додатку середовища Mathlab є можливість використовувати згорткові нейронні мережі і мережі довгострокової пам'яті для виконання класифікації та регресії зображень, часових рядів і текстових даних. Також створювати мережеві архітектури, такі як генеративні змагальні мережі і сіамські мережі, використовуючи автоматичне диференціювання, користувацькі цикли навчання та спільні ваги.

За допомогою додатка Deep Network Designer можна проектувати, аналізувати та навчати мережі графічно. Програма Experiment Manager допоможе керувати кількома експериментами глибокого навчання, відстежувати параметри навчання, аналізувати результати та порівнювати код із різних експериментів. Є можливість візуалізувати активацію шарів і графічно відстежувати хід навчання.

Можливий обмін моделями з TensorFlow і PyTorch через формат ONNX та імпортувати моделі з TensorFlow-Keras і Caffe. Набір інструментів підтримує передачу навчання з DarkNet-53, ResNet-50, NASNet, SqueezeNet та багатьма іншими попередньо підготовленими моделями [11].

В даному додатку є можливість створювати нові глибокі мережі для завдань класифікації зображень і регресії, визначаючи архітектуру мережі та навчаючи мережу з нуля. А також використовувати навчання з перенесенням, щоб скористатися знаннями, наданими попередньо підготовленою мережею, для вивчення нових шаблонів у нових даних. Точне налаштування попередньо вивченої мережі класифікації зображень із перенесенням навчання, як правило, набагато швидше та легше, ніж навчання з нуля. Використання попередньо навчених глибоких мереж дає змогу швидко вивчати нові завдання без визначення та навчання нової мережі, не маючи мільйонів зображень або потужного графічного процесора [11].

## 2.4 Вибір та обґрунтування програмного забезпечення для реалізації ШНМ

Серед вище перелічених програмних забезпечень для реалізації штучної нейронної мережі, яка буде розпізнавати потрібні компоненти друкованого модуля серед масиву поданих зображень на вхід, обираємо MathLab. Через додаток Deep Learning toolbox є можливість візуально створювати та редагувати нейронну мережу потрібної нами архітектури. Використання даного програмного забезпечення не потребує глибоких знань програмування для реалізації архітектури НМ як DarkNet-53, ResNet-50, NASNet, SqueezeNet. Також надає можливість візуальної перевірки на сприйняття поданих вхідних даних для навчання мережі і порівняння, отримання результатів у табличній та графічній формі.

# 3 ВИБІР АРХІТЕКТУРИ НЕЙРОННОЇ МЕРЕЖІ

Для створення нейронних мереж використовуються різні типи архітектур, на рисунку 3.1 наведені архітектури по популярності, нейронних мереж, які мають свої переваги та недоліки. Для реалізації ШНМ для поставленої задачі розглянемо найпоширеніші архітектури нейронних мереж.

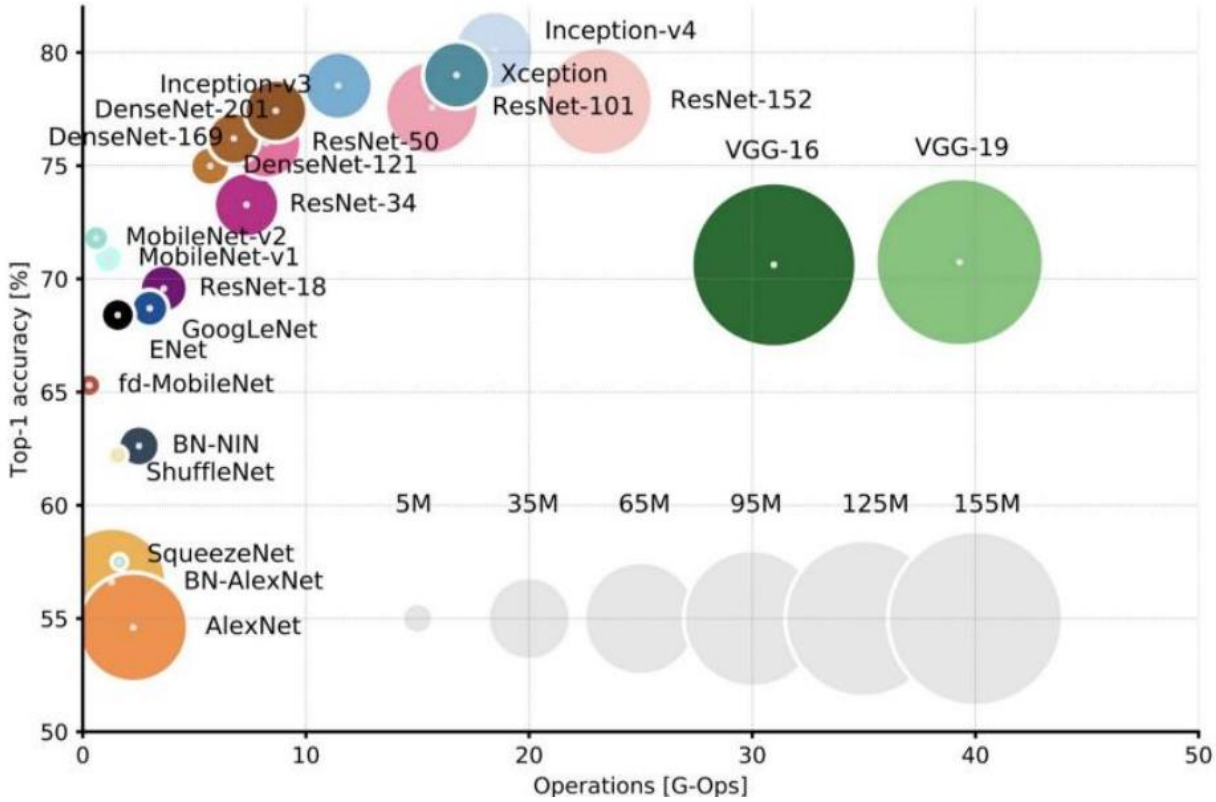


Рисунок 3.1 — Порівняння популярних архітектур НМ

В даній магістерській дисертації розглянуто 3 класи нейронних мереж, а саме згорткові, рекурентні та прямого поширення, та розглянуто типи нейронних мереж в хронологічному порядку [2].

## 3.1 Огляд архітектур

У глибокому навчанні згорткова нейронні мережі — це клас штучних нейронних мереж, найчастіше використовуваних аналізу візуальних зображень. Вони також відомі як інваріантні до зсуву або просторово-інваріантні штучні нейронні мережі, засновані на архітектурі спільної ваги згорткових ядер або фільтрів та забезпечують еквівалентні до перекладу відповіді, відомі як карти ознак. Як не дивно, більшість згорткових нейронних мереж еквівалентні, а не інваріантні. Вони знаходять застосування у розпізнаванні зображень і відео, рекомендаційних системах, класифікації зображень, сегментації зображень, аналізі медичних зображень, обробці природної мови, інтерфейс мозок-комп'ютер та фінансових часових рядах [4].

Згорткові нейронні мережі є регуляризовані версії багатошарових перцептронів. Під багатошаровими перцептронами зазвичай маються на увазі пов'язані мережі,

тобто кожен нейрон в одному шарі пов'язаний з усіма нейронами в наступному шарі. Повна зв'язність цих мереж робить їх схильними до переоцінки даних. Типові способи регуляризації, або запобігання перепідгонки, включають: штрафування параметрів під час навчання (наприклад, загасання ваги) або обрізання зв'язності згорткові нейронні мережі використовують інший підхід до регуляризації: вони використовують переваги ієрархічної структури даних і збирають схеми зростаючої складності, використовуючи дрібніші та простіші схеми, закладені в їх фільтрах, тому на шкалі зв'язності та складності знаходяться на нижньому кордоні [3].

Рекурентна нейронна мережа — це один з класів штучної нейронної мережі і використовується в областях застосування природного опрацювання мови та розпізнавання мовлення. Модель RNN розроблена для розпізнавання послідовних характеристик даних і надалі з використанням шаблонів для прогнозування майбутнього сценарію [4].

Переваги періодичних нейронних мереж:

- RNN може обробляти входи будь-якої довжини.
- Модель RNN змодельована для запам'ятовування кожної інформації протягом усього часу, що дуже корисно для будь-якого прогнозованого часового ряду.
- Навіть якщо розмір вводу більший, розмір моделі не збільшується.
- Ваги можна розділити протягом кроків часу.
- RNN може використовувати свою внутрішню пам’ять для обробки довільних рядів входів, що не стосується подальших нейронних мереж.

Недоліки періодичних нейронних мереж:

- Через свою періодичність, обчислення відбуваються повільно.
- Навчання моделей RNN може бути важким.
- Якщо ми використовуємо relu або tanh як функції активації, обробляти дуже довгі послідовності стає дуже важко.
- Схильний до таких проблем, як вибух та зникнення градієнта.

Нейронна мережа з прямою передачею – це штучна нейронна мережа, в якій зв'язки між вузлами не утворюють циклу. Цим вона відрізняється від свого нащадка — рекурентних нейронних мереж.

Нейронна мережа з прямою передачею була першим і найпростішим типом штучних нейронних мереж. У цій мережі інформація рухається лише в одному напрямку - вперед від вхідних вузлів, через приховані вузли (якщо вони є) та до вихідних вузлів. У мережі немає циклів чи петель.

Далі будуть розглянуті типи архітектур нейронних мереж по збільшенню точності навчання та хронології виникнення [5].

Архітектура нейронної мережі LeNet — згорткова нейронна мережа, що використовує послідовність з трьох шарів: шари згортки (convolution), шари групування (pooling) та шари нелінійності (non-linearity). У 1994-му була розроблена одна з перших згорткових нейромереж, яка започаткувала глибоке навчання. Ця піонерська робота Яна Лекуна (Yann LeCun) після багатьох успішних ітерацій, починаючи з 1988-го, отримала назву LeNet5.

Архітектура LeNet5 стала фундаментальною для глибокого навчання, особливо з погляду розподілу властивостей зображення по всій картинці. Згортки з параметрами, що навчаються, дозволяли за допомогою декількох параметрів ефективно отримувати однакові властивості з різних місць. Ключовою перевагою архітектури виявилася можливість зберігати параметри та результати обчислень, на відміну від використання кожного пікселя як окремі вхідні дані для великої багатошарової нейромережі. У LeNet5 у першому шарі пікселі не використовуються, тому що зображення сильно скорельовано просторово, так що використання окремих пікселів як вхідних властивостей не дозволить скористатися перевагами цих кореляцій.

Особливості LeNet5:

- Використовує згортку для отримання просторових властивостей.
- Підвибірка із використанням просторового усереднення карт.
- Нелінійність у вигляді гіперболічного тангенсу або сигмоїду.
- Фінальний класифікатор у вигляді багатошарової нейромережі.
- Розріджена матриця зв'язності між шарами дозволяє зменшити обсяг обчислень.

У 2012-му Олексій Крижевський опублікував AlexNet, поглиблену та розширену версію LeNet, яка з великим відривом перемогла у складному змаганні ImageNet.

В AlexNet результати обчислень LeNet масштабовані в набагато більшу нейромережу, яка здатна вивчити набагато складніші об'єкти та їх ієрархії. Особливості цього рішення:

- Використання блоків лінійної ректифікації (ReLU) як нелінійності.
- Використання методики відкидання для вибіркового ігнорування окремих нейронів під час навчання, що дозволяє уникнути перенавчання моделі.
- Перекриття max pooling, що дозволяє уникнути ефектів усереднення average pooling.
- Використання NVIDIA GTX 580 для прискорення навчання.

На той час кількість ядер у відеокартах сильно зросла, що дозволило приблизно в 10 разів скоротити час навчання, і в результаті стало можливо використовувати куди більші датсети і картинки [3].

Успіх AlexNet запустив велику революцію, згорткові нейронні мережі перетворилися на робочу конячку глибокого навчання — цей термін відтепер означав "великі нейромережі, здатні вирішувати корисні завдання".

У розроблених в Оксфорді VGG-мережах у кожному згортковому шарі вперше застосували фільтри 3x3 та ще й об'єднали ці шари в послідовності згорток.

Це суперечить закладеним у LeNet принципам, згідно з якими великі згортки використовувалися для отримання однакових властивостей зображення. Замість застосовуваних AlexNet фільтрів 9x9 і 11x11 стали застосовувати набагато дрібніші фільтри, небезпечно близькі до згорток 1x1, яких намагалися уникнути автори LeNet, принаймні в перших шарах мережі. Але великою перевагою VGG стала знахідка, що кілька згорток 3x3, об'єднаних у послідовність, можуть емулювати великі рецептивні поля, наприклад, 5x5 або 7x7.

Мережі VGG для уявлення складних властивостей використовують численні згорткові шари 3x3. Але вчити такі мережі було складно, доводилося розбивати їх на дрібніші, додаючи шари один за одним. Причина полягала у відсутності ефективних способів регуляризації моделей або методів обмеження великого простору пошуку, якому сприяє безліч параметрів.

VGG у багатьох шарах використовують велику кількість властивостей, тому навчання вимагає великих обчислювальних витрат. Зменшити навантаження можна зменшивши кількість властивостей, як це зроблено в bottleneck-шарах архітектури Inception.

В основі архітектури Network-in-network (NiN) лежить проста ідея: використання згорток 1x1 для збільшення комбінаторності властивостей у згорткових шарах.

У NiN після кожного згортки застосовуються просторові MLP-шари, щоб краще комбінувати властивості перед подачею до наступного шару. Може здатися, що використання згорток 1x1 суперечить вихідним принципам LeNet, але це дозволяє комбінувати властивості краще, ніж просто набиваючи більше згорткових шарів. Цей підхід відрізняється від використання голих пікселів як вхідні дані для наступного шару. У цьому випадку згортки 1x1 застосовуються для просторового комбінування властивостей після згортки в рамках карток властивостей, так що можна використовувати набагато менше параметрів, які є спільними для всіх пікселів цих властивостей.

MLP дозволяють сильно підвищити ефективність окремих згорткових шарів за допомогою їх комбінування більш складні групи. Ця ідея пізніше була використана в інших архітектурах, таких як ResNet, Inception та їх варіантах.

Крістіан Жегеді (Christian Szegedy) з Google перейнявся зниженням обсягу обчислень у глибоких нейромережах, і в результаті створив GoogLeNet — першу архітектуру Inception.

До осені 2014-го моделі глибокого навчання стали дуже корисними у кластеризації вмісту зображень та кадрів з відео. Багато скептиків визнали користь глибокого навчання і нейромереж, а інтернет-гіганти, у тому числі Google, дуже зацікавилися розгортанням на своїх серверних потужностях ефективних і великих мереж.

Крістіан шукав шляхи зменшення обчислювального навантаження в нейромережах, домагаючись високої продуктивності (наприклад, ImageNet). Або зберігаючи обсяг обчислень, але при цьому підвищуючи продуктивність.

В результаті команда створила модуль Inception. На перший погляд, це паралельна комбінація згорткових фільтрів 1x1, 3x3 та 5x5. Але особливість полягала у використанні згорткових блоків 1x1 (NiN) зменшення кількості властивостей перед подачею в «дорогі» паралельні блоки. Зазвичай цю частину називають bottleneck.

У GoogLeNet як початковий шар застосовується stem без Inception-модулів, а також використовується average pooling і softmax-класифікатор, аналогічний NiN. Цей класифікатор виконує дуже мало операцій порівняно з AlexNet та VGG. Це також допомогло створити дуже ефективну архітектуру нейромережі, як Bottleneck-

шар. Цей шар зменшує кількість властивостей (а значить і операцій) у кожному шарі, тому швидкість отримання результату можна зберегти на високому рівні. Перш ніж передавати дані в "дорогі" згорткові модулі, кількість властивостей зменшується, скажімо, в 4 рази. Це сильно скорочує обсяг обчислень, що забезпечило архітектурі популярність.

Останні дослідження в галузі глибоких нейронних мереж були спрямовані переважно на підвищення точності. Для заданого рівня точності, як правило, можна виділити кілька архітектур DNN, які досягають цього рівня точності. При еквівалентній точності менші архітектури DNN мають принаймні три переваги:

- Маленькі DNN вимагають менше обміну даними між серверами під час розподіленого навчання.
- Для експорту нової моделі з хмари в автономний автомобіль меншим DNN потрібна менша пропускна спроможність.
- Маленькі DNN більш доцільно розгортати на ПЛІС та іншому обладнанні з обмеженою пам'яттю.

Всі особливості вищеперелічених архітектур комбіновані в дуже ефективну і компактну мережу, що використовує дуже мало властивостей і обчислювальних потужностей, але при цьому видає чудові результати. Архітектура отримала назву ENet, її розробив Адам Пазке (Adam Paszke).

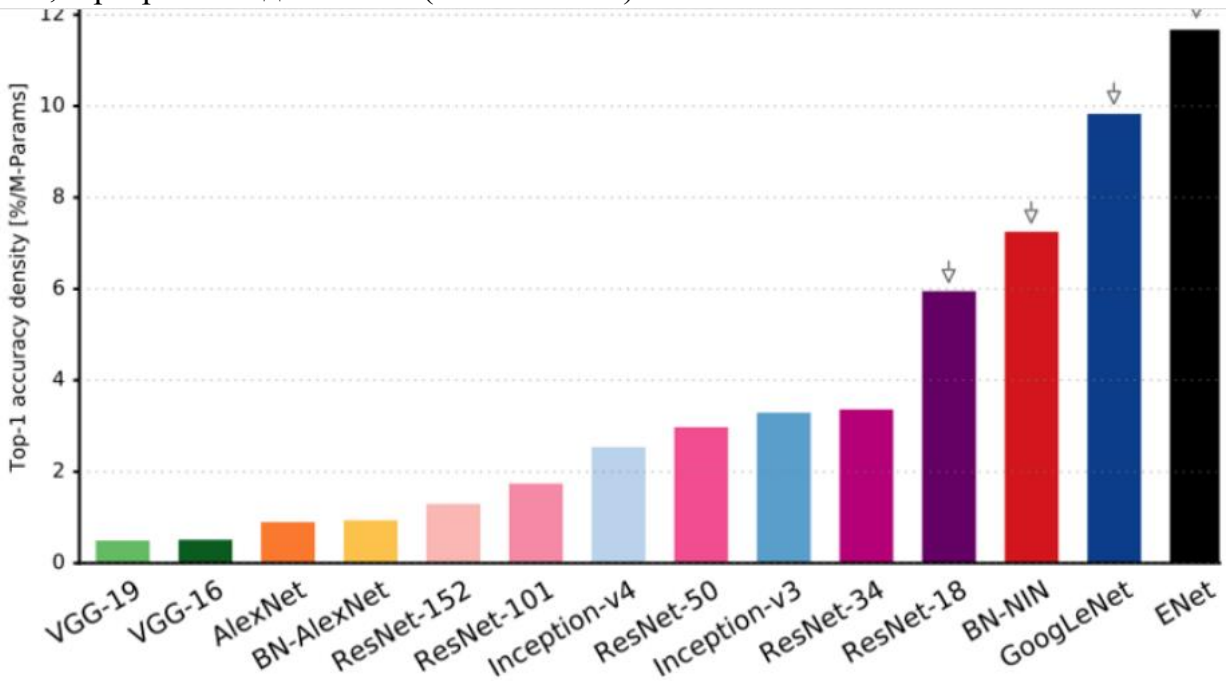


Рисунок 3.2 — Питома точність мережі Enet у порівнянні з іншими мережами

Це мережа на основі кодувальника та декодера. Кодувальник побудований за звичайною схемою CNN для категоризації, а декодер є мережею з підвищенням дискретизації (upsampling network), призначену для сегментування за допомогою поширення категорій назад у зображення вихідного розміру. Для сегментації зображень використовувалися лише нейромережі, жодних інших алгоритмів.

ENet створювалася з розрахунку, щоб від початку використовувати якнайменше ресурсів. В результаті копіювач і декодер разом займають лише 0,7 Мб з точністю

fp16. І при такому крихітному розмірі ENet за точністю сегментування не поступається або перевершує інші суто нейромережні рішення [2].

## 3.2 Вибір архітектури нейронної мережі

Згорткові нейронні мережі (ЗНМ) — це клас штучних нейронних мереж прямого поширення, який застосовується для аналізу зображень. ЗНМ використовують різновид багатошарових перцептронів, розроблений так, щоб вимагати використання мінімального обробки. Виходячи з їхньої архітектури спільних ваг та характеристик інваріантності відносно паралельного перенесення. ЗНМ використовують порівняно мало попередньої обробки, в порівнянні з іншими алгоритмами класифікування зображень. Це означає, що мережа навчається за допомогою фільтрів, що в традиційних алгоритмах приходиться розробляти вручну. Ця незалежність у конструюванні ознак від апріорних знань та людських зусиль є великою перевагою.

ЗНМ складається з шарів входу та виходу, а також із декількох прихованих шарів. Приховані шари ЗНМ зазвичай складаються зі згорткових шарів, агрегувальних шарів, повноз'єднаних шарів та шарів нормалізації. Згорткові шари застосовують до входу операцію згортки, передаючи результат до наступного шару. Згортка імітує реакцію окремого нейрону на зоровий стимул.

Для розпізнання компонентів електронного модулю будемо використовувати тип згорткової мережі squeezenet.

SqueezeNet — це нейронна мережа, яка була розроблена як компактна заміна AlexNet. Вона має майже в 50 разів менше параметрів, ніж AlexNet, але при цьому працює втричі швидше.

Основні ідеї, що лежать в основі SqueezeNet:

- Стратегія перша: використовувати фільтри 1×1 замість 3×3
- Стратегія друга: зменшити кількість вхідних каналів до 3×3 фільтрів
- Стратегія третя: Зменшення дискретизації на пізніх етапах мережі, щоб шари згортки мали великі карти активації.

Архітектура SqueezeNet складається з шарів "squeeze" та "expand". Стиснутий згортковий шар має тільки фільтри 1 × 1. Вони надходять у розширюючий шар, який має суміш згорткових фільтрів 1 × 1 і 3 × 3. Це показано нижче на рисунку 3.3

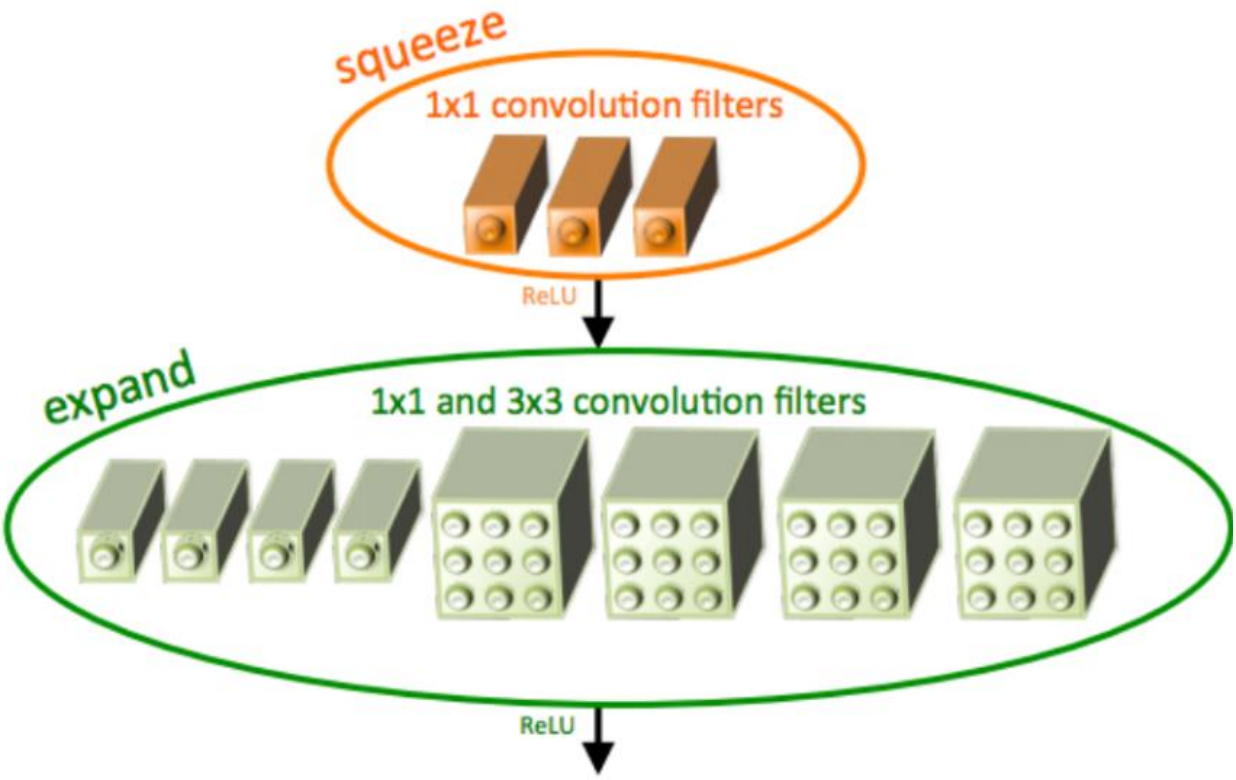


Рисунок 3.3 — Шари із яких складається мережа SqueezeNet

Вхідне зображення спочатку надходить окремий коеволюційний шар. За цим шаром слідують 8 "вогненних модулів", які називаються "fire2-9", відповідно до стратегії 1, описаної вище (рис. 3.4).

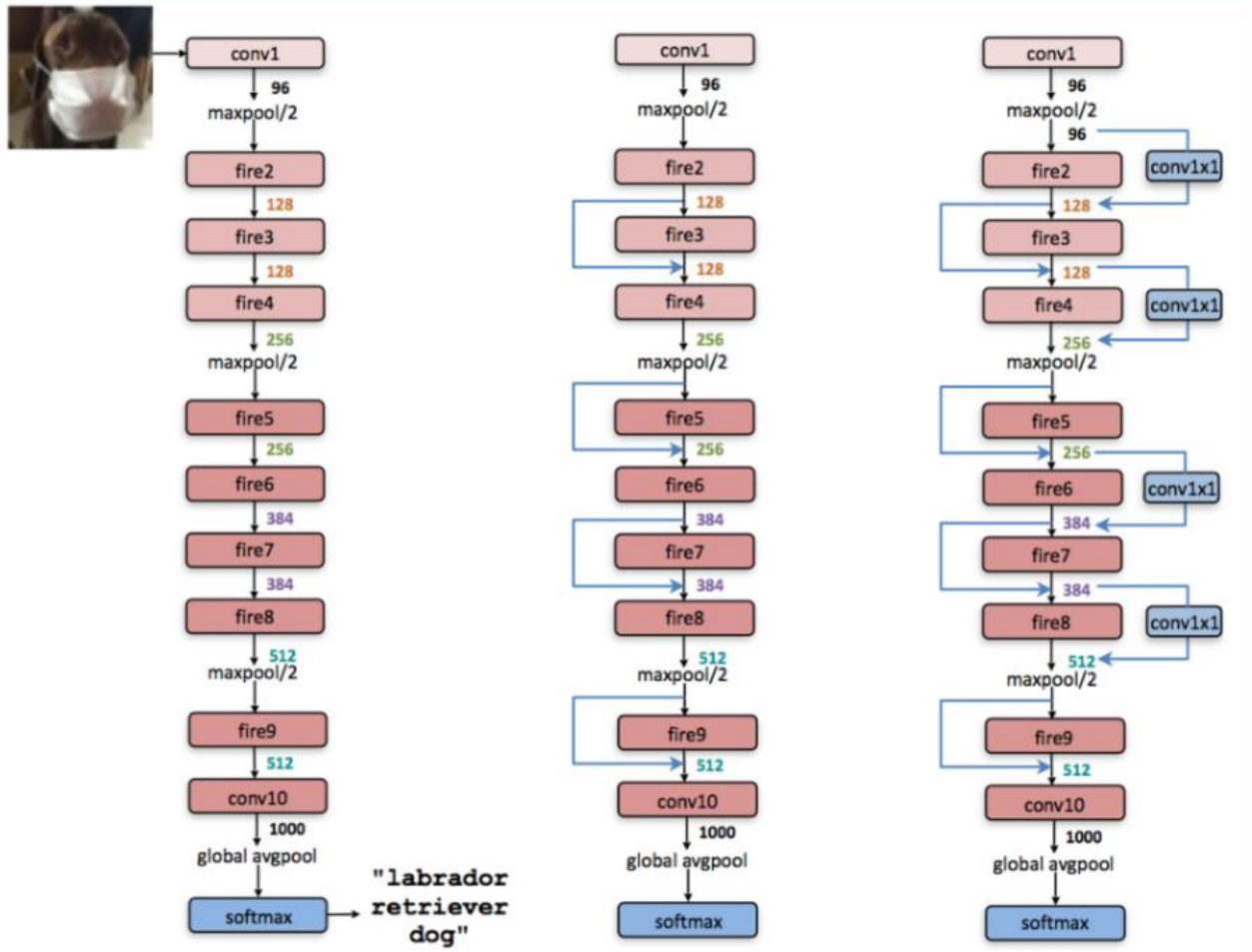


Рисунок 3.4 — Слої мережі C

Згідно з другою стратегією, фільтри на модуль fire збільшуються за допомогою "простого обходу". SqueezeNet виконує max-pooling з кроком 2 після шарів conv1, fire4, fire8 та conv10. Відповідно до стратегії три, pooling розміщується відносно пізно, що призводить до SqueezeNet зі "складним обходом".

Нижче наведено подробиці щодо інших параметрів, що використовуються в мережі:

- Активація ReLU застосовується між усіма шарами стиснення та розширення всередині модуля Fire.
- Для зменшення надлишкової підгонки додаються шари, що відсіюють, з ймовірністю 0,5 після модуля fire9.
- У мережі не використовуються пов'язані шари. Цей вибір дизайну надихнув архітектура Network In Network (NIN), запропонована (Lin et al, 2013).
- SqueezeNet навчалася із швидкістю навчання 0,04, яка лінійно зменшується протягом усього процесу навчання.
- Розмір партії для навчання складає 32 і мережа використовувала оптимізатор Адама.
- SqueezeNet полегшує процес розгортання завдяки своєму невеликому розміру. Спочатку ця мережа була реалізована в Caffe, але з того часу

модель набула популярності і була впроваджена на безліч різних платформ.

Отже дана архітектура нейронної мережі підходить для реалізації поставленої задачі, а саме для розпізнавання зображень основних компонентів електронного модулю — резисторів, транзисторів, конденсаторів, діодів та мікросхем. Також дана архітектура НМ підходить для пошуку дефектів в електронних модулів шляхом порівняння еталонного зображення компонентів із зображенням яке поступає на вхід.

# 4 РЕАЛІЗАЦІЯ ШТУЧНОЇ НЕЙРОННОЇ МЕРЕЖІ

Перевірку можливості використання обраної нейронної мережі в задачах пошуку дефектів електронного модулю проведемо на прикладі побудови класифікатора електронних компонентів, які встановлюються на друковану плату. Для цього виконаємо наступні дії:

- Підготовка даних для навчання
- Навчання нейронної мережі
- Тестування НМ

## 4.1 Підготовка даних для навчання

Основним завданням для підготовки будь якої нейронної мережі є формування датасету. Датасет — набір вихідних даних на основі яких нейронна мережа проводить навчання. В цій роботі датасет складається із 5 класів елементів: резисторів, транзисторів, конденсаторів, діодів та інтегральних мікросхем(ІМС). Приклад даних зображень наведений на рис. 4.1. Кожен клас складається з 360 зображень. Оскільки Matlab не сприймає зображення із різним розширенням та розміром дані зображення повинні бути однакового розміру та розширення.

Рисунок 4.1 — Зображення діода, резистора, конденсатора та транзистора SMD

Для даного датасету обрані зображення із розширенням 49x31 пікселів, яке відповідає найменшому зображенню, та стиснуті до розміру 1кБайт. На рис. 4.2 показано кількість зображень в кожної із класів, які подаються на вхід нейронної мережі.

```
>> NM

tbl =

  5×2 table

       Label        Count
    ____________    _____

    capacity         360
    diodes           360
    ims              360
    resistors        360
    tranzisrors      360
```

Рисунок 4.2 — Кількість елементів в кожному класі.

Створені об’єкти командами:

path = fullfile(fileparts(mfilename('fullpath')),'\dataset');

imds =imageDatastore (path,'IncludeSubfolders',1,'LabelSource','foldernames');

Де в змінні imds знаходиться посилання на зображення які будемо використовувати.

Функція 'IncludeSubfolders' включає підпапки які знаходяться в основной папці із зображеннями.

Функції 'LabelSource','foldernames' використовуються для встановлення міток на кожному із зображенні, щоб мітки відповідали до назви папок.

## 4.2 Навчання нейронної мережі

Для створення згорткової мережі типу SqueezeNet використовуємо DeepLearning toolbox , де вибираємо потрібний нам тип та робимо аналіз НМ.

Як можемо бачити із рис. 4.3 наша НМ складається із 68 шарів на вхід яких подається зображення розміром 227x227x3.

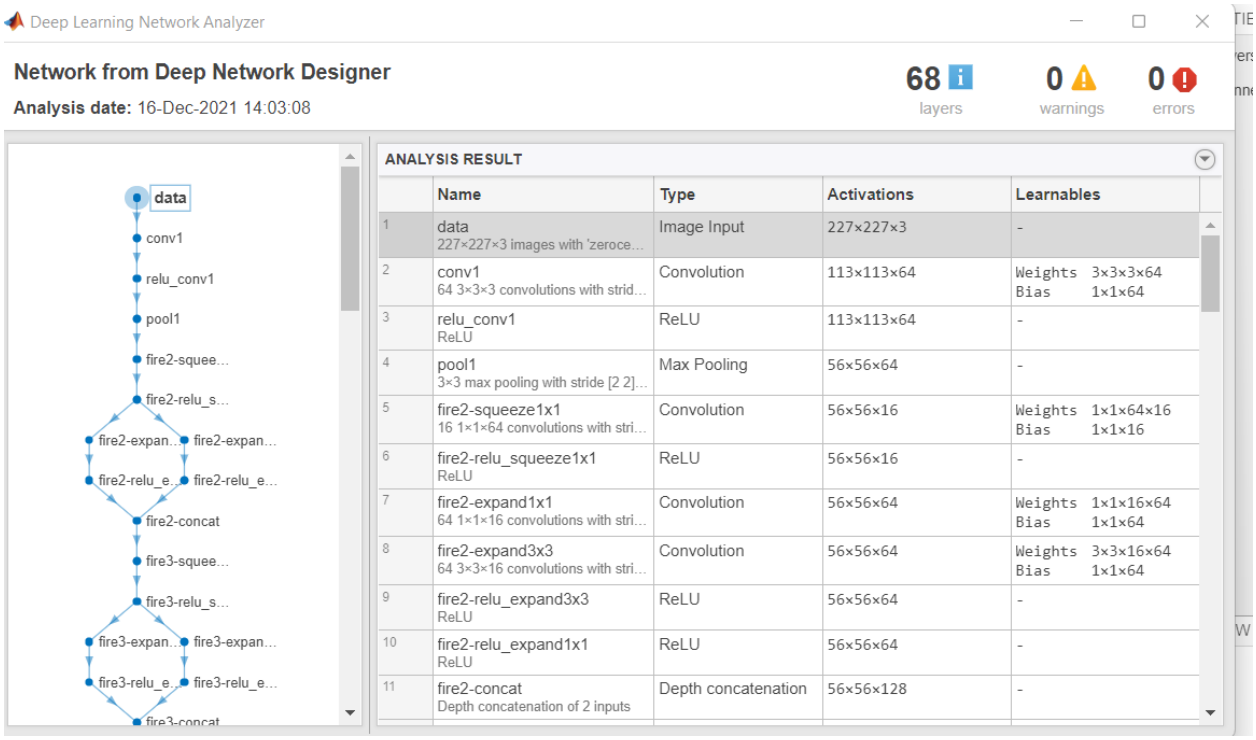


Рисунок 4.3 — Аналіз НМ SqueezeNet.

Перед початком навчання НМ потрібно замінити останній згортковий шар на шар із новими параметрами під які буде реалізована наша НМ. А саме на вхід повинно подаватись згортка із 5 класів елементів відповідно замінюємо кількість фільтрів на 5, збільшуємо швидкість навчання в 10 разів, також заміняємо вихідний шар класифікатора на такий ж шар класифікатора тільки для потрібних нам 5 класів (рис. 4.4).

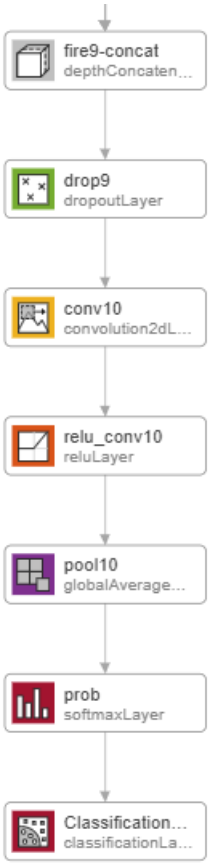


Рисунок 4.4 — Останні шар нейронної мережі SqueezeNet

Для того щоб почати навчання мережі потрібно передати створену мережу на навчання та задати опції навчання, а саме вибрати алгоритм, вибрати швидкість навчання, задати кількість епох, задати розмір пакету який буде передаватись на навчання. Від цих параметрів буде залежати точність класифікації мережі.

Функція ReadFunction опис якої наведено нижче, потрібна для того щоб перетворити розмір вхідного зображення до розміру який сприймає наша НМ.

```
function I = readFunction(filename)
I = imread(filename);
I = imresize(I,[227 227]);
```

Наступною функцією ми розподіляємо наші зображення на тренувальну вибірку і на навчальну виборці і функцією 'randomized' розподілу їх випадковим чином.

```
[train, test] = imds.splitEachLabel(0.6, 'randomized');
```

Задаємо параметри нашої НМ:

```
layers = [...
    imageInputLayer([31 49 1]),...
    convolution2dLayer(5,20),...
    reluLayer,...
    maxPooling2dLayer(2,"Stride",2),...
    fullyConnectedLayer(5),...
    softmaxLayer,...
    classificationLayer];
```

Та передаєм їх для навчання:

options = trainingOptions('sgdm', 'InitialLearnRate', 0.0005, 'MaxEpochs',50, 'Plots','training-progress');

net = trainNetwork(train, layers, options);

Після того як ми передали наші зображення на навчання і задали параметри навчання НМ можемо отримати результат навчання.

Таблиця 4. 1 — Результати навчання НМ за 15 епох.

| Epoch | Iteration | Time Elapsed | Mini-batch | Mini-batch | Base Learning |
|---|---|---|---|---|---|
| 1 | 1 | 00:00:2 | 2,56% | 2.7487 | 0.0005 |
| 15 | 15 | 00:00:14 | 76,45% | 0.6941 | 0.0005 |

Як можемо бачити на рис. 4.5 показаний результат навчання НМ при встановленій кількості епох – 15, швидкості навчання – 0,0005 та кількості зображень в кожному класі по 30 елементів. Точність навчання складає 73,33% тому що дана НМ розрахована на більшу кількість навчальної вибірки.

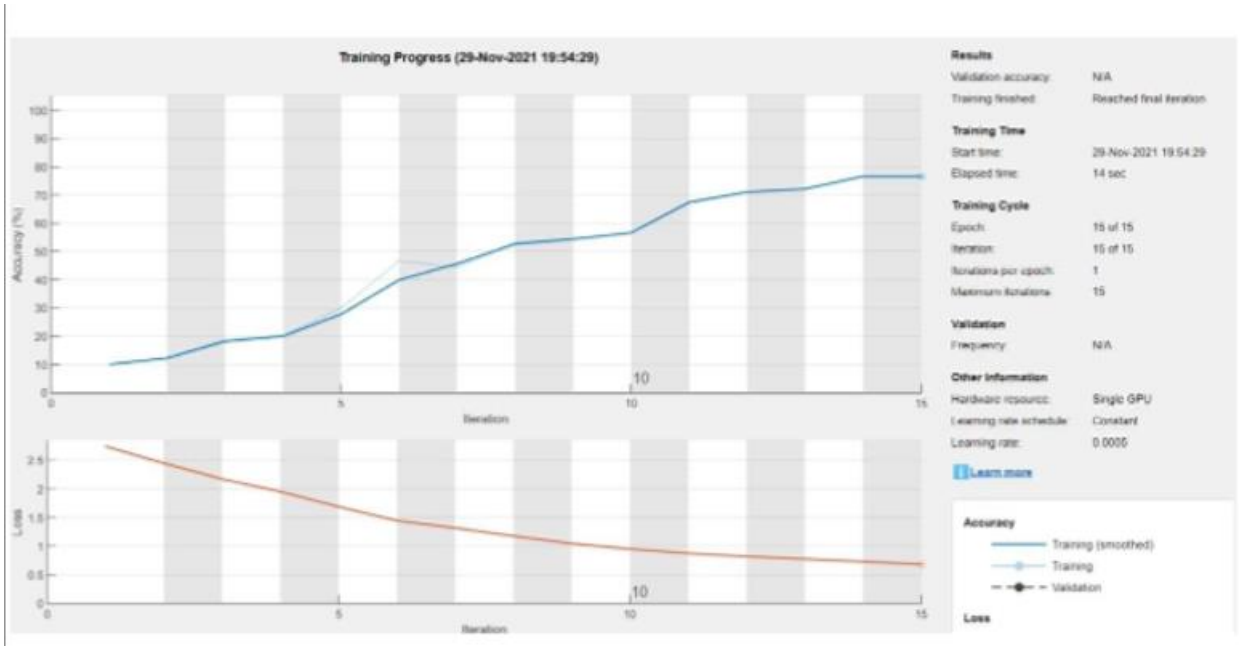


Рисунок 4.5 — Графік навчання НМ

Щоб досягти більшої точності навчання, змінимо кількість епох до 50 та кількість навчальної вибірки збільшимо до 360 зображень на кожен із класів та проведемо навчання.

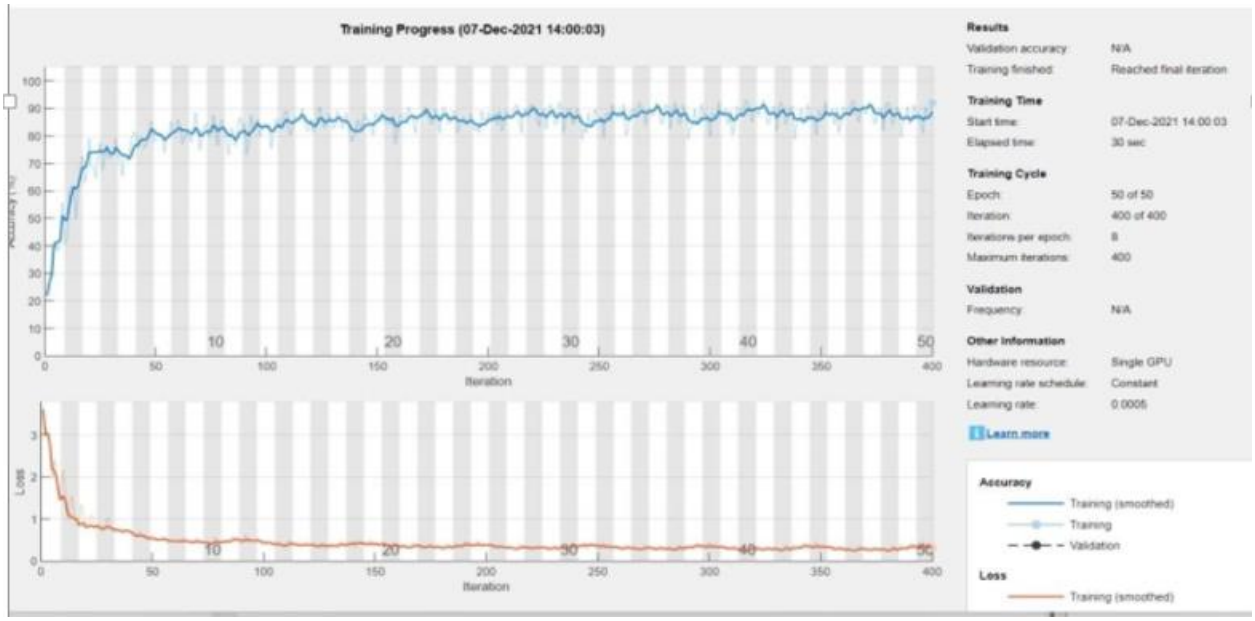


Рисунок 4.6 — результат навчання НМ при зміні кількості зображень та епох

З рисунку 4.6 можемо побачити що при збільшенні кількості епох та зображень в кожному із класів збільшиться точність.

Для отримання більш високої точності потрібно зрозуміти які елементи зображення не сприймає матлаб та спробувати замінити їх.

### 4.3 Тестування нейронної мережі

Щоб дослідити чому нейронна мережа показала низьку точність можна перевірити які зображення з датасету вона не сприймає. Зробити це можливо за допомогою функції:

test.Files(t~=test.Labels)

Виконавши дану функцію отримуємо результат:

{'C:\Users\Admin\Desktop\HM\dataset\capacity\diodes_8.png' }
{'C:\Users\Admin\Desktop\HM\dataset\capacity\diodes_82 (2).png' }
{'C:\Users\Admin\Desktop\HM\dataset\capacity\diodes_82 (3).png' }
{'C:\Users\Admin\Desktop\HM\dataset\capacity\diodes_82 (2).png' }
{'C:\Users\Admin\Desktop\HM\dataset\capacity\diodes_82 png'

Що свідчить про те що в навчальній вибірці, а саме в папці “Capacity” знаходяться зображення діодів, тому нейронна мережа не може правильно провести навчання, оскільки в класі конденсатори нейронна мережа бачить діоди і в класі діоди вважає їх конденсаторами і через це низький відсоток точності розпізнавання компонентів.

Після того як виправили даних дефект в навчальної виборки можемо проаналізувати отримані результати в таблиці 4.2 та рисунку 4.7. Як бачимо із нижче наведених даних наша нейронна мережа має приблизно 96% точності розпізнавання основних компонентів електронного модулю.

Таблиця 4.2 — Результат тестування нейронної мережі.

| Epoch | Iteration | Time Elapsed | Mini-batch | Mini-batch | Base Learning |
|---|---|---|---|---|---|
| 1 | 1 | 00:00:22 | 19.53% | 3.5487 | 0.0005 |
| 7 | 50 | 00:00:31 | 89.06% | 0.3341 | 0.0005 |
| 13 | 100 | 00:00:35 | 96.09% | 0.1576 | 0.0005 |
| 19 | 150 | 00:00:40 | 97.66% | 0.1122 | 0.0005 |
| 25 | 200 | 00:00:45 | 97.66% | 0.0884 | 0.0005 |
| 32 | 250 | 00:00:50 | 97.66% | 0.0982 | 0.0005 |
| 38 | 300 | 00:00:55 | 100.00% | 0.0475 | 0.0005 |
| 44 | 350 | 00:01:00 | 99.22% | 0.0514 | 0.0005 |
| 50 | 400 | 00:01:04 | 100.00% | 0.0349 | 0.0005 |

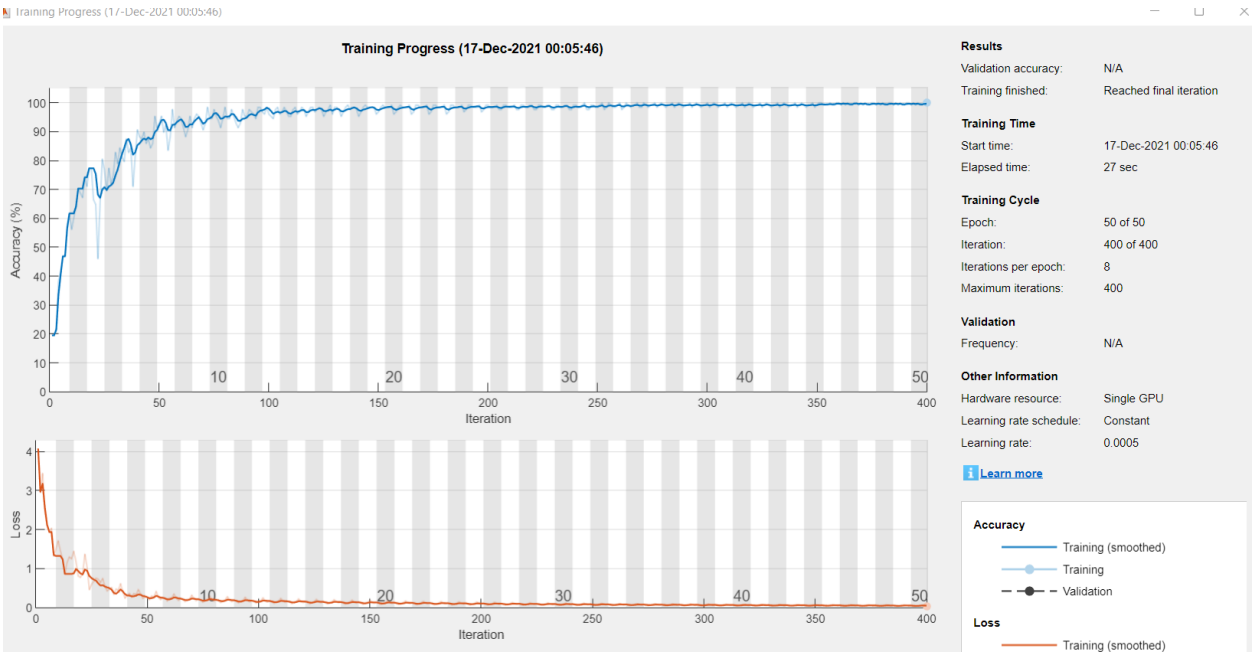


Рисунок 4.7 — Результат навчання нейронної мережі

Повторно застосувавши функцію t(t~=test.Labels) побачимо, що нейронна мережа не сприймає деякі зображення, такі як:

{'C:\Users\rusa2\Desktop\NM\dataset\capacity\capacitors_18 (3).png' }
{'C:\Users\rusa2\Desktop\NM\dataset\capacity\capacitors_18.png' }
{'C:\Users\rusa2\Desktop\NM\dataset\capacity\capacitors_19 (4).png }
{'C:\Users\rusa2\Desktop\NM\dataset\capacity\capacitors_2 (22).png' }
{'C:\Users\rusa2\Desktop\NM\dataset\capacity\capacitors_2 (3).png' }

На рисунку 4.8 показані ці зображення, які нейронна мережа не сприймає можливо через те що погана якість зображення і НМ сприймає ці елементи як елементи інших класів: резисторів, діодів, тощо.

Рисунок 4.7 — Зображення конденсаторів із навчальної вибірки

Проте незважаючи на наявність хибних класифікацій все ж нейронна мережа здатна 96% випадків правильно класифікувати електронний компонент на друкованій платі.

Таким чином, ми отримали нейронну мережу яка з високою достовірністю проводить класифікацію електронних компонентів, встановлених на друковану плату. В цілому це підтверджує успішність застосування нейронних мереж в задачах пошуку дефектів електронного модулю.

Одним з варіанті застосування мережі може бути проведення швидкої класифікації електронного компоненту після застосування алгоритму кластеризації зображення електронного модулю. Тобто після поділу зображення на кластери за допомогою даної нейронної мережі можна провести визначення типу компоненту в кожному кластері і вже далі проводити визначення наявності чи відсутності дефекту.

# ВИСНОВКИ

Проведений аналіз типових дефектів, які виникають в процесі виготовлення електронних модулів, та методів їх виявлення показав, що зараз, серед іншого, використовуються візуальні методи контролю, які проходять за участі людини. Запропоновано використати штучні нейронні мережі для зменшення навантаженості оператора шляхом зменшення кількості хибних спрацювань комп'ютерного алгоритму виявлення дефекту.

В якості програмного забезпечення для реалізації ШНМ обрано MatLab, так як його використання не потребує специфічних знань в програмуванні. Крім того, він містить вбудований модуль для роботи з передовими архітектурами нейронних мереж — Deep Learning Toolbox.

Огляд типових архітектур нейронних мереж дозволив обрати SqueezeNet, як базову архітектуру для вирішення задачі класифікації електронних компонентів, як частини алгоритму пошуку дефектів на електронних модулях.

В процесі роботи над дисертацією підготовлено Датасет зображень електронних компонентів, який складається з 5 категорій: резистори, конденсатори, діоди, транзистори та мікросхеми. Кожна категорія містить 360 зображень відповідних електронних компонентів.

Тестування нейронної мережі показало, що точність достовірного визначення типу електронного компоненту, встановленого на друковану плату, складає 96%, що є високим результатом.

# ПЕРЕЛІК ДЖЕРЕЛ ТА ПОСИЛАНЬ

1. 8 Common Errors in Surface Mount Technology [Електронний ресурс] – Режим доступу до ресурсу: https://www.protoexpress.com/blog/common-errors-surface-mount-technology-smt

2. The Essential Guide to Neural Network Architectures [Електронний ресурс] – Режим доступу до ресурсу: https://www.v7labs.com/blog/neural-network-architectures-guide

3. Повторні нейронні мережі [Електронний ресурс] – Режим доступу до ресурсу: https://uk.education-wiki.com/4082704-recurrent-neural-networks-rnn.

4. Сверточная нейронная сеть [Електронний ресурс] – Режим доступу до ресурсу: https://evergreens.com.ua/ru/articles/cnn.html.

5. НЕЙРОННЫЕ СЕТИ НА PYTHON [Електронний ресурс] – Режим доступу до ресурсу: https://kpfu.ru/portal/docs/F_1458204831/Nejronnye.seti.na.Python.pdf.

6. Получение простой нейронной сети для работы с нуля в C++ [Електронний ресурс] – Режим доступу до ресурсу: https://coderoad.ru

7. OpenNN [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/OpenNN

8. Deep Learning Toolbox [Електронний ресурс] – Режим доступу до ресурсу:https://www.mathworks.com/help/deeplearning/index.html?s_tid=CRUX_lftnav

# ДОДАТОК А

```
clear all

%% Data
path = fullfile(fileparts(mfilename('fullpath')),'\dataset');
imds = imageDatastore(path,'IncludeSubfolders',1,'LabelSource','foldernames');

tbl = countEachLabel(imds)

%%
rng(0)

imds = splitEachLabel(imds,1);
imds.ReadFcn = @readFunction;
%%
imds = imageDatastore(path, 'IncludeSubfolders',true,'LabelSource', 'foldernames');
[train, test] = imds.splitEachLabel(0.6, 'randomized');
layers = [...
   imageInputLayer([31 49 1]),...
   convolution2dLayer(5,20),...
   reluLayer,...
   maxPooling2dLayer(2,"Stride",2),...
   fullyConnectedLayer(5),...
   softmaxLayer,...
   classificationLayer];

options = trainingOptions('sgdm', 'InitialLearnRate', 0.0005, 'MaxEpochs',50,
'Plots','training-progress');
net = trainNetwork(train, layers, options);

t = classify(net, test);
sum(t==test.Labels)/numel(t)
test.Files(t~=test.Labels)
```