РОЗДІЛ 6

ПРОГРАМНІ ЗАСОБИ ДЛЯ МОДЕЛЮВАННЯ НЕЙРОМЕРЕЖ

6.1 Моделювання нейронних мереж у пакеті Matlab

Matlab являє собою математичний пакет, призначений для вирішення задач обчислювальної математики та побудови чисельних моделей складних об'єктів і процесів.

Пакет Matlab складається з інтерпретатора – модельного середовища, що має термінальний інтерфейс, ядра (набору найпростіших стандартних операцій, функцій і процедур для обчислень), а також бібліотек функцій (Toolbox).

Перевагами пакета є: багаті графічні можливості, великий набір математичних функцій, простота вбудованої мови Matlab, можливість автоматичного перетворення текстів програм мовою Matlab у тексти програм на мові Сі, а також те, що тексти бібліотечних функцій постачаються у вихідному виді.

До недоліків пакета варто віднести низьку швидкість роботи.

Для моделювання НМ за допомогою пакета Matlab необхідно встановити і використовувати бібліотеку *Deep Learning Toolbox* (раніше *–Neural Network Toolbox*). Бібліотека містить як функції для моделювання мілких (shallow networks), так і глибоких (deep networks) нейромереж.

Пакет Matlab, починаючи з версії 6.0, містить візуальний інтерфейсний модуль *nntool*, який входить до бібліотеки Neural Network Toolbox. Використання nntool дозволяє більш зручними засобами, ніж написання програми вручну, будувати нейромережеві моделі. Розглянемо деякі основні можливості та прийоми роботи із засобом nntool.

Після запуску Matlab. exe в командному вікні для початку роботи із nntool треба ввести: `nntool`. Після цього завантажиться засіб nntool (рис. 6.1).

На панелі Network and Data («Нейромережі та дані») користувач має натиснути кнопки для завдання вихідних даних для побудови нейромережевої моделі.

Кнопка New Data («Нові дані») викликає редактор для створення нових даних (рис. 6.2).

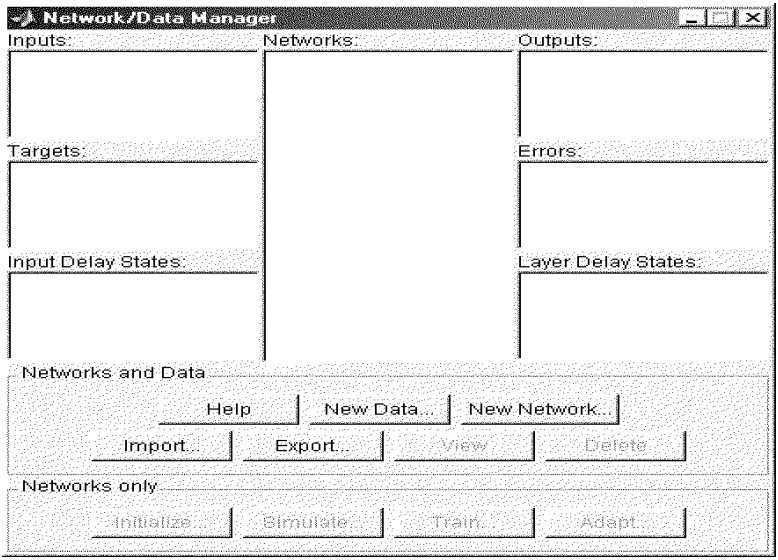


Рисунок 6.1 – Головна діалогова форма засобу nntool

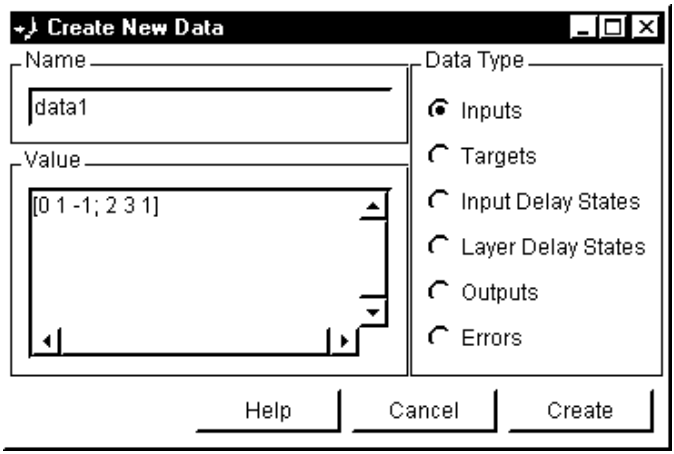


Рисунок 6.2 – Редактор даних засобу nntool

Поле Name («Ім’я») задає ім’я нової змінної середовища Matlab, до якої зберігається масив нових даних, що вводяться у полі Value («Значення»).

Панель Data Type («Тип даних») визначає призначення введених даних: Inputs – входи мережі, Targets – цільові значення виходів мережі, Input Delay States та Layer Delay States – описи затримок на входах та у шарах мережі, Outputs – фактичні значення на виходах мережі, Errors – помилки мережі.

Якщо дані вже існують у вигляді зовнішніх файлів або містяться у середовищі Matlab у вигляді змінних, вони можуть бути імпортовані за допомогою кнопки Import («Імпорт»). При цьому з'являється діалогова форма (рис. 6.3).

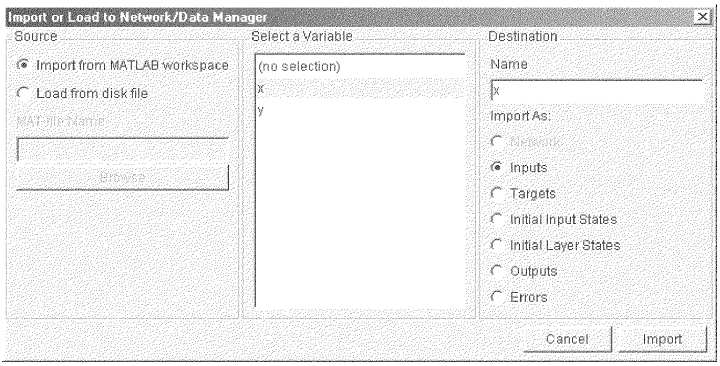


Рисунок 6.3 – Діалогова форма імпорту даних

Поле Source («Джерело») дозволяє обрати джерело введення даних: Import from Matlab workspace (імпорт даних із середовища Matlab) або Load from disk file (завантаження даних із файлу на диску). Кнопка Browse дозволяє обрати необхідний файл.

Поле Select a Variable («Вибір змінної») дозволяє вказати засобу nntool, яку змінну треба використовувати для імпорту даних.

Панель Destination («Приймач») дозволяє задати змінну для прийому даних, що імпортуються. Її ім'я вказується у полі Name («Ім'я»), а призначення (Import as) обирається із наведеного меню.

Кнопка Export («Експорт») головної діалогової форми дозволяє зберегти дані із середовища nntool у файлі на диску, або передати їх до середовища Matlab.

Кнопка «New Network» («Нова мережа») викликає діалогову форму для конструювання нейромережі та визначення її параметрів (рис. 6.4).

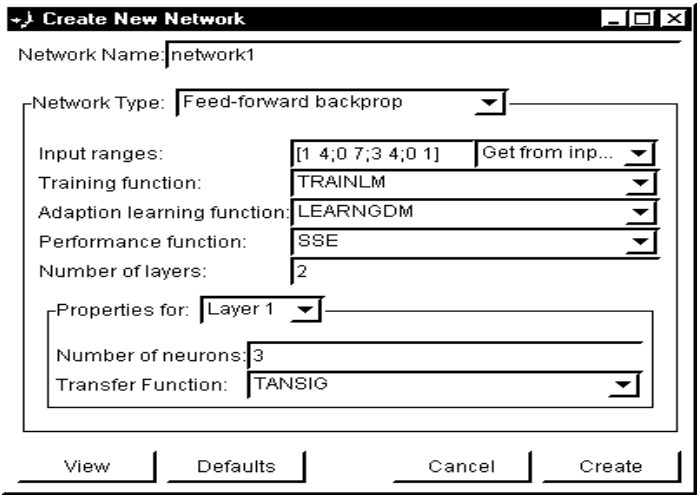


Рисунок 6.4 – Форма конструювання нейромережі

Поле Network Name («Ім'я мережі») визначає ім'я змінної, де зберігається мережа. Список вибору Network Type («Тип мережі») дозволяє обрати тип архітектури мережі (наприклад, Feed-forward backprop – багатошарова нейромережа прямого поширення), поля Training Function, Adaptation Learning Function, Performance Function та Number of Layers визначають, відповідно, тип методу навчання мережі, тип методу адаптації ваг мережі, цільову функцію та кількість шарів мережі.

Панель Properties for Layer K дозволяє задати властивості для нейронів K-го шару мережі. У полі Number of Neurons вказують кількість нейронів для поточного шару, а у полі Transfer Function – тип функції активації нейронів поточного шару мережі.

Кнопка «View» дозволяє отримати графічне зображення схеми побудованої нейромережі (рис. 6.5).

Кнопка «Delete» головної форми дозволяє видалити непотрібний елемент даних (змінну або мережу).

Кнопка «Help» дозволяє викликати довідкову службу Matlab з описом необхідних компонентів та поясненнями щодо їхнього використання.

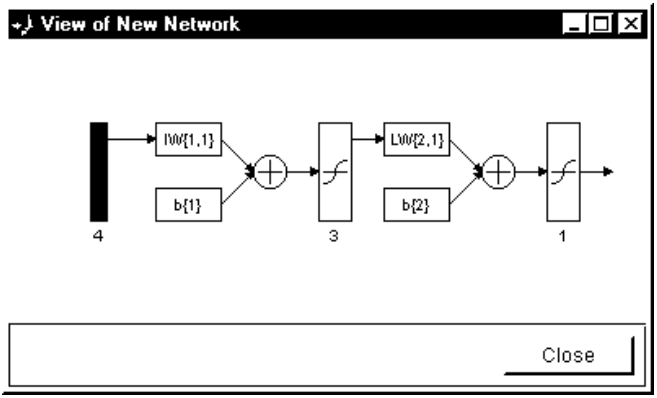


Рисунок 6.5 – Приклад зображення форми нейромережі, побудованої за допомогою засобу nntool

Після побудови нейромережі у нижній частині головної діалогової форми nntool стає доступною панель Networks only, що призначена для роботи із побудованою мережею (рис. 6.6).

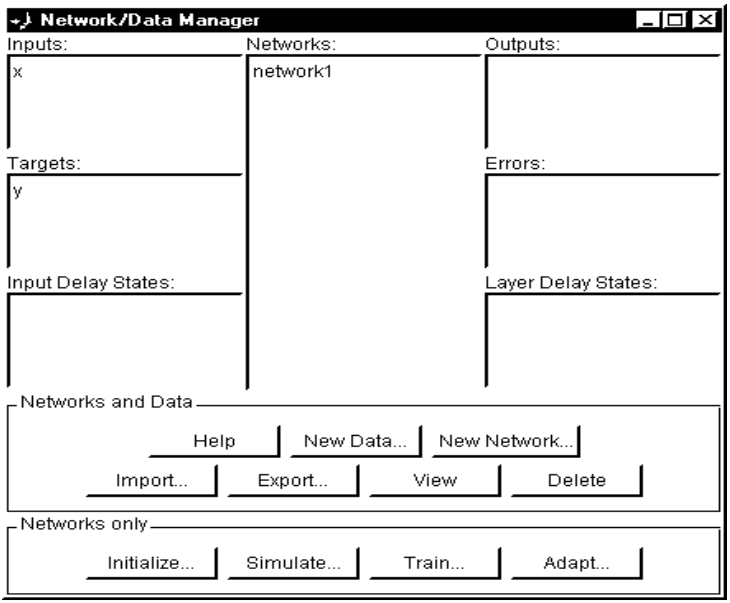


Рисунок 6.6 – Головна діалогова форма nntool після побудови мережі

При натисненні будь-якої з кнопок цієї панелі викликається діалогова форма Network («Мережа»), яка містить набір закладок-панелей для роботи з мережею (рис. 6.7–6.10).

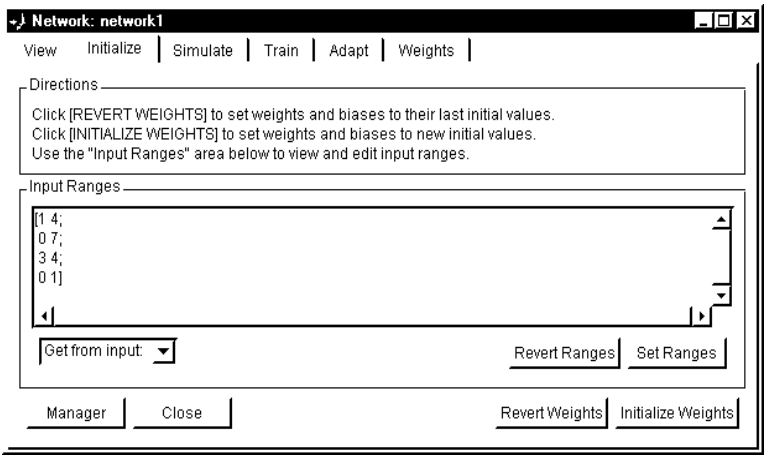


Рисунок 6.7 – Форма Network: закладка Initialize

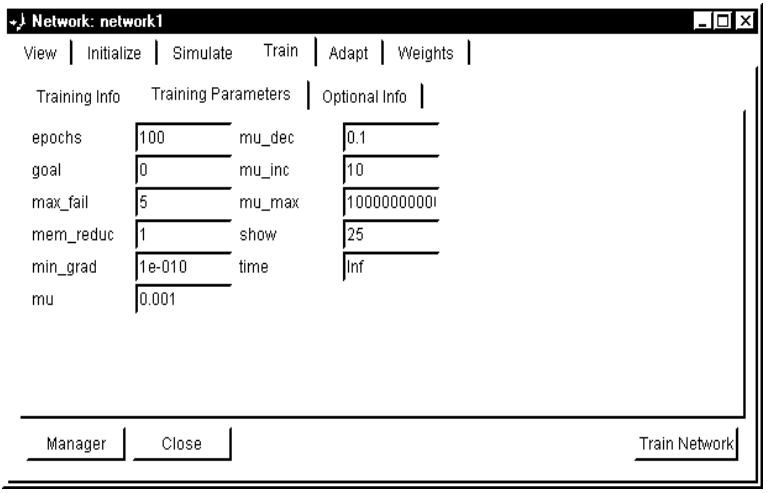


Рисунок 6.8 – Форма Network: закладка Train

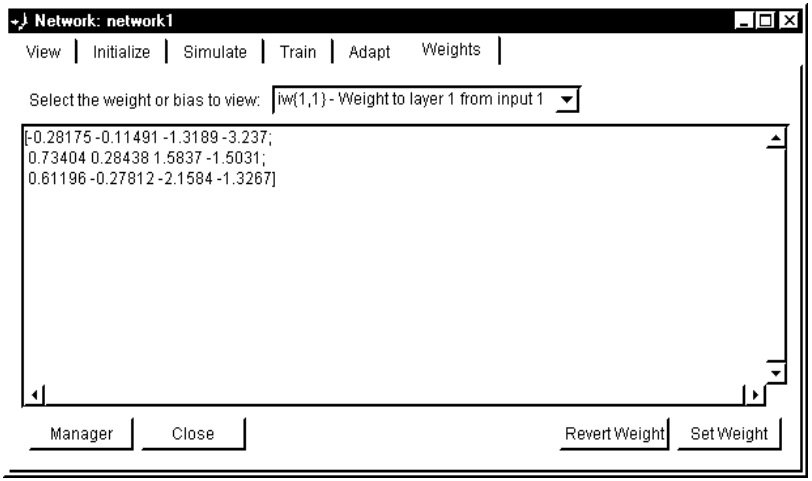


Рисунок 6.9 – Форма Network: закладка Weights

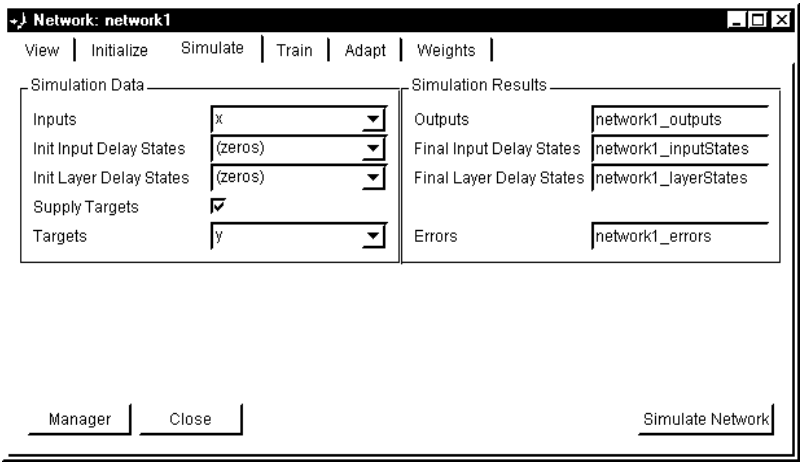


Рисунок 6.10 – Форма Network: закладка Simulate

Закладка Initialize («Ініціалізація») дозволяє задати межі, в яких змінюються вхідні дані та розрахувати на їхній основі початкові значення ваг мережі.

Ініціалізована мережа може бути навчена за допомогою закладки Train («Тренування, навчання»). Серед параметрів

навчання, доступних на цій закладці обов'язково необхідно задати: goal – максимально припустиме значення цільової функції, epochs – максимальна припустима кількість циклів навчання мережі, show – крок виведення на екран інформації про навчання мережі, задається у циклах навчання.

У процесі навчання середовище Matlab будує графік зміни значення цільової функції за епохами – циклами навчання (рис. 6.11).

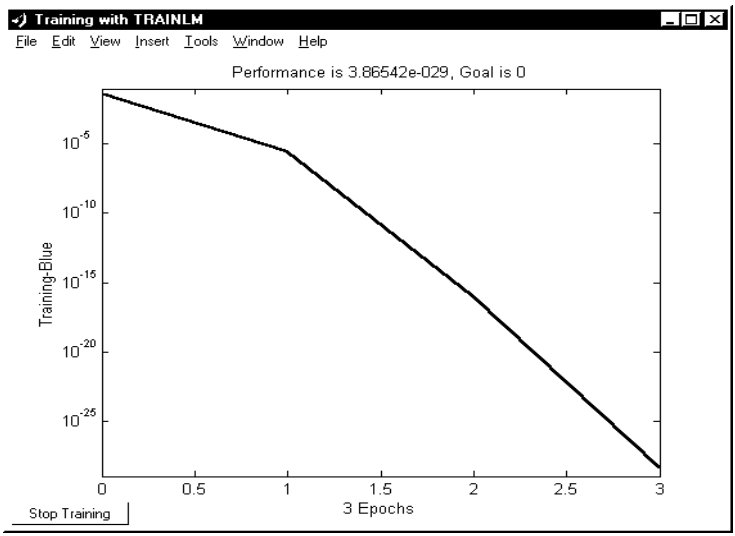


Рисунок 6.11 – Графік зміни значення цільової функції в процесі навчання

Ваги мережі, що була навчена, можна переглянути, використовуючи закладку Weights («Ваги»).

Після того, як мережа навчилася, її можна використовувати для розпізнавання за допомогою закладки Simulate («Моделювання»).

Нейронні мережі, створені за допомогою функцій Matlab, мають єдиний формат подання, як це показано у табл. 6.1.

Поряд із використанням засобу nntool пакет Matlab дозволяє використовувати функції для моделювання нейромереж у програмному та ручному режимах.

Таблиця 6.1 – Подання структури нейромережі у пакеті Matlab

Поле структури	Опис, значення поля
numInputs	кількість входів мережі
numLayers	кількість (схованих) шарів мережі без урахування вхідного шару
biasConnect, inputConnect, layerConnect, outputConnect, targetConnect	булеві масиви, що визначають зв'язки між структурними елементами мережі
numOutputs	кількість виходів мережі
numTargets	кількість цільових ознак
numInputDelays	кількість затримок вхідного шару
numLayerDelays	кількість затримок схованих шарів
inputs	входи мережі
layers	шари мережі
outputs	виходи мережі
targets	цільові ознаки
biases	масив порогів нейронів мережі
inputWeights	масив вагових коефіцієнтів вхідного шару мережі
layerWeights	масив вагових коефіцієнтів схованих шарів мережі
adaptFcn	ім'я функції адаптації нейронів
initFcn	ім'я функції ініціалізації мережі
performFcn	ім'я цільової функції навчання (помилки)
trainFcn	ім'я функції, що реалізує процес навчання
adaptParam	параметри адаптації мережі
initParam	параметри ініціалізації мережі
performParam	параметри цільової функції мережі
trainParam. epochs	максимально допустима кількість ітерацій навчання (епох)
trainParam.. goal	максимально допустиме значення цільової функції навчання
trainParam. max_fail	максимально допустима кількість відмов у процесі навчання мережі
trainParam. mem_reduc	коефіцієнт, що регулює (зменшує) використання пам'яті при навчанні нейромереж
trainParam. min_grad	мінімально допустиме значення градієнту цільової функції
trainParam. mu, trainParam. mu_dec, trainParam. mu_inc, trainParam. mu_max	параметри методу Левенберга-Марквардта
trainParam. show	кількість ітерацій, через яку будуть відображатися зміни стану процесу навчання мережі
trainParam. time	максимально допустимий час навчання мережі у секундах
IW	масив значень ваг вхідного (першого) шару
LW	масив значень ваг схованих шарів
b	масив значень порогів нейронів
userdata	дані користувача

Перелік найважливіших функцій наведено в табл. 6.2. Формат виклику функцій можна дізнатися, вводячи команду: `help ім'я_функції`.

Таблиця 6.2 – Основні функції, використовувані для моделювання та побудови нейромереж у пакеті Matlab

Група функцій	Ім'я функції	Призначення або результат роботи функції
1	2	3
Функції для аналізу	rrsurf	Поверхня помилки нейрона з єдиним входом
	maxlinlr	Максимальна швидкість навчання для лінійного нейрона
Функції відстані	boxdist	Відстань між двома векторами
	dist	Евклідова функція
	linkdist	Зв'язана функция
	mandist	Манхеттенська метрика
Функції навчання	learncon	Навчальна функція порогів
	learngd	Навчальна функція градієнтного спуску
	learngdm	Навчальна функція градієнтного спуску з урахуванням моментів
	learnh	Навчальна функція Хебба
	learnhd	Навчальна функція Хебба з урахуванням загасання
	learnis	Навчальна функція instar
	learnk	Навчальна функція Кохонена
	learnlv1	Навчальна функція LVQ1
	learnlv2	Навчальна функція LVQ2
	learnos	Навчальна функція outstar
	learnp	Навчальна функція ваг і порогів персептрона
	learnpn	Навчальна функція нормалізованих порогів і ваг персептрона
	learnsom	Навчальна функція SOM
	learnwh	Правило навчання Уідроу-Хоффа
Функції графіки	hintonw	Графік Хінтона для матриці ваг
	hintonwb	Графік Хінтона для матриці ваг і векторів порогів
	plotbr	Графік функціонування мережі при регулярному байєсовському тренуванні
	plotep	Зображення положень ваг і порогів на поверхні помилки
	plotes	Зображення поверхні помилок одиничного вхідного нейрона

Продовження таблиці 6.2

1	2	3
	plotpc	Зображення лінії класифікації у векторному просторі персептрона
	plotperf	Графічне подання функціонування мережі
	plotpv	Графічне подання вхідних цільових векторів
	plotsom	Графічне подання SOM
	plotv	Графічне подання векторів у виді ліній, що виходять з початку координат
	plotvec	Графічне подання векторів різними кольорами
Топологічні функції	gridtop	Топологічна функція у виді сіткового шару
	hextop	Топологічна функція у виді гексагонального шару
	randtop	Топологічна функція у виді випадкового шару
Лінійні функції пошуку	srchbac	Одновимірна мінімізація з використанням пошуку з поверненням
	srchbre	Одновимірна локалізація інтервалу з використанням методу Брента
	srchcha	Одновимірна мінімізація з використанням методу Караламбуса
	srchgol	Одновимірна мінімізація з використанням золотого перетину
	srchhyb	Одновимірна мінімізація з використанням гібридного бісекційного пошуку
Функції використання мережі	adapt	Адаптація мережі
	disp	Відображає властивості нейронної мережі
	display	Відображає імена змінних і властивості мережі
	init	Ініціалізація нейронної мережі
	sim	Моделювання нейронної мережі
	train	Тренування нейронної мережі
	nntool	Виклик графічного інтерфейсу користувача
	gensim	Генерація блоку Simulink для моделювання нейронної мережі
Функції створення нової мережі	network	Створення нейронної мережі користувача
	newc	Створення конкурентного шару
	newcf	Створення каскадної спрямованої мережі
	newelm	Створення мережі Елмана
	newff	Створення мережі прямого поширення
	newfftd	Створення прямого поширення з вхідними затримками
	newgrnn	Створення узагальненої регресійної нейронної мережі

Продовження таблиці 6.2

1	2	3
	newhop	Створення мережі Хопфілда
	newlin	Створення лінійного шару
	newlind	Конструювання лінійного шару
	newlvq	Створення мережі LVQ
	newp	Створення персептрона
	newpnn	Конструювання імовірнісної нейронної мережі
	newrb	Конструювання радіально-базисної мережі
	newrbe	Конструювання точної мережі з радіальними базисними функціями
	newsom	Створення мережі SOM
Цільові функції і їхні похідні	dmae	Похідна функції mae
	dmse	Похідна функції mse
	dmsereg	Похідна функції msereg
	dsse	Похідна функції sse
	mae	Середня абсолютна помилка
	mse	Середньоквадратична помилка
	msereg	Зважена сума середньоквадратичної помилки і середнього квадратів значень ваг і порогів
	sse	Сумарна квадратична помилка
Функції попередньої і пост- обробки	postmnmx	Ненормалізовані дані, що були нормалізовані за допомогою prenmmx
	postreg	Лінійний регресійний аналіз виходів мережі стосовно цільових значень навчального масиву
	poststd	Ненормовані дані, що були нормовані за допомогою функції prestd
	premnmx	Нормування даних у діапазоні від 1 до +1
	prepca	Аналіз головних компонентів для вхідних даних
	prestd	Нормування даних до одиничного стандартного відхилення і нульовому середній
	tramnmx	Перетворення даних з попередньо обчисленими мінімумом і максимумом
	trapca	Перетворення даних з використанням PCA-матриці (головних компонент), обчисленої за допомогою функції prepca
	trastd	Перетворення даних з використанням попередньо обчислених значень стандартного відхилення і середнього
Функції навчання (тренування)	trainb	Пакетне тренування з використанням правил навчання для ваг і порогів

Продовження таблиці 6.2

1	2	3
	trainbfg	Тренування мережі з використанням квазіньютонівського методу Бройдена-Флетчера-Гольдфарба-Шенно
	trainbr	Регуляризація Байєса
	trainc	Використання збільшень циклічного порядку
	traincgb	Метод зв'язаних градієнтів Пауелла-Біла
	traincgf	Метод зв'язаних градієнтів Флетчера-Пауелла
	traincgp	Метод зв'язаних градієнтів Полака-Рібьєра
	traingd	Метод градієнтного спуску
	traingda	Метод градієнтного спуску з адаптивним навчанням
	traingdm	Метод градієнтного спуску з урахуванням моментів
	traingdx	Метод градієнтного спуску з урахуванням моментів і адаптивним навчанням
	trainlm	Метод Левенберга-Марквардта
	trainoss	Одноступінчатий метод січних
	trainr	Метод випадкових збільшень
	trainrp	Метод пружного зворотного поширення
	trains	Метод послідовних збільшень
	trainscg	Метод шкальованих зв'язаних градієнтів
Вагові функції і їхні похідні	dnetprod	Обчислення похідної від входів мережі з перемножуванням входів
	dnetsum	Обчислення похідної від входів мережі з підсумовуванням входів
	netprod	Функція добутку входів
	netsum	Функція підсумовування входів
	ddotprod	Похідна скалярного добутку
	dist	Евклідова відстань
	dotprod	Вагова функція у виді скалярного добутку
	mandist	Вагова функція відстань Манхеттена
	negdist	Вагова функція негативна відстань
	normprod	Вагова функція нормований скалярний добуток
Функції активації і їхні похідні	dhardlim	Похідна східчастої функції активації
	dhardlms	Похідна симетричної східчастої функції активації
	dlogsig	Похідна сигмоидної (логістичної) функції активації
	dposlin	Похідна позитивної лінійної функції активації
	dpurelin	Похідна лінійної функції активації
	dradbas	Похідна радіально-базисної функції активації
	dsatlin	Похідна лінійної функції активації з насиченням

Продовження таблиці 6.2

1	2	3
	dsatlins	Похідна симетричної функції активації з насиченням
	dtansig	Похідна функції активації гіперболічний тангенс
	dtribas	Похідна трикутної функції активації
	compet	Конкуруюча функція активації
	hardlim	Східчаста функція активації
	hardlims	Східчаста симетрична функція активації
	logsig	Сигмоїдна (логістична) функція активації
	poslin	Позитивна лінійна функція активації
	purelin	Лінійна функція активації
	radbas	Радіально-базисна функція активації
	satlin	Лінійна функції активації з насиченням
	satlins	Симетрична лінійна функція активації, що насичується
	softmax	Функція активації, що зменшує діапазон вхідних значень
	tansig	Функція активації гіперболічний тангенс
	tribas	Трикутна функція активації
Допоміжні функції	calca	Обчислює виходи мережі й інші сигнали
	calca1	Обчислює сигнали мережі для одного кроку за часом
	calce	Обчислює помилки шарів
	calce1	Обчислює помилки шарів для одного кроку за часом
	calcgx	Обчислює градієнт ваг і порогів як єдиний вектор
	calcjejj	Обчислює Якобіан
	calcjx	Обчислює Якобіан ваг і порогів як одну матрицю
	calcpd	Обчислює затримані входи мережі
	calcperf	Обчислення виходів мережі, сигналів і функціонування
	formx	Формує один вектор з ваг і порогів
	getx	Повертає усі ваги і пороги мережі як один вектор
	setx	Установлює усі ваги і пороги мережі у виді одного вектора
	cell2mat	Поєднує масив елементів матриць в одну матрицю
	combvec	Створює всі комбінації векторів

Продовження таблиці 6.2

1	2	3
	con2seq	Перетворить вектори, що сходяться, у послідовні вектори
	concur	Створює вектори порогів, що сходяться
	ind2vec	Перетворення індексів у вектори
	mat2cell	Розбивка матриці на масив елементів матриць
	minmax	Обчислює мінімальні і максимальні значення рядків матриці
	normc	Нормує стовпці матриці
	normr	Нормує рядки матриці
	pnormc	Псевдо-нормування стовпців матриці
	quant	Дискретизація величини
	seq2con	Перетворення послідовних векторів у вектори, що сходяться
	sumsqr	Сума квадратів елементів матриці
	vec2ind	Перетворення векторів в індекси
Функції ініціалізації ваг і порогів	initcon	Функція ініціалізації порогів
	initzero	Ініціалізація з установкою нульових значень ваг і порогів
	midpoint	Ініціалізація з установкою середніх значень ваг
	randnc	Ініціалізація з установкою нормалізованих значень стовпців вагових матриць
	randnr	Ініціалізація з установкою нормалізованих значень рядків вагових функцій
	rands	Ініціалізація з установкою симетричних випадкових значень ваг і порогів
	revert	Повернення вагам і порогам значень, що відповідають попередньої ініціалізації
	initnw	Функція ініціалізації Нгуена-Уідроу
	initwb	Функція ініціалізації по вагам і порогам
	initlay	Функція пошарової ініціалізації мережі

Приклади використання функцій пакету Matlab наведено у підрозділі 6.4.

Поряд із використанням функцій модуля Neural Network Toolbox для автоматизації певних етапів синтезу НМ у пакеті Matlab можна використовувати функції інших модулів.

Особливе місце серед інших функцій посідають *функції вирішення задач оптимізації на основі еволюційного пошуку*, які містить бібліотека *Genetic Algorithm and Direct Search Toolbox* пакету Matlab.

Для використання генетичних методів, у середовищі Matlab передбачена функція [*x*, Fmin] = ga(@fitnessfun, n, options), яка знаходить мінімум Fmin функції fitnessfun, що має n параметрів, а також вектор x, при якому досягається мінімум цільової функції fitnessfun.

Параметри роботи функції ga задаються у змінній options, що являє собою структуру, у якій можуть бути задані: спосіб подання інформації у хромосомі, використовувані генетичні оператори відбору, схрещування й мутації, критерії зупинення й інші параметри генетичного пошуку. За допомогою структури options також можна вибрати графіки, які будуть відображати основні результати генетичного пошуку в процесі оптимізації цільової функції.

Для зміни значень параметрів функції ga передбачена функція gaoptimset, а для одержання поточних параметрів – функція gaoptimget.

Моделювання глибоких нейромереж у пакеті Matlab можна здійснити шляхом використання засобу *Deep Network Designer*. Він забезпечує дружній інтерфейс користувача для конструювання, візуалізації, навчання, редагування глибоких нейромереж. Також засіб дозволяє зберігати та завантажувати нейромоделі, копіювати, видаляти та редагувати їхні складові, а також аналізувати мережу, імпортувати дані, налаштовувати параметри мереж та методів навчання, слідкувати за точністю.

Для запуску засобу потрібно увести команду: `deepNetworkDesigner`.

Форми засобу DeepNetworkDesigner наведені на рис. 6.12–6.18.

Імпорт даних у Deep Network Designer для навчання здійснюють таким чином. Спочатку на вкладці "Дані" треба натиснути "Імпортувати дані". Можна імпортувати дані з папки з підпапками зображень для кожного класу або з imageDatastore в робочій області. Deep Network Designer надає вибір варіантів збільшення зображень. Можна ефективно збільшити обсяг навчальних даних, застосувавши до даних рандомізоване збільшення. Якщо треба збільшити дані, Deep Network Designer випадковим чином вносить зміни до даних на кожній епосі. Кожна епоха потім використовує дещо інший набір даних.

Імпорт тестових даних здійснюють, обравши папку або імпортуючи imageDatastore з робочої області.

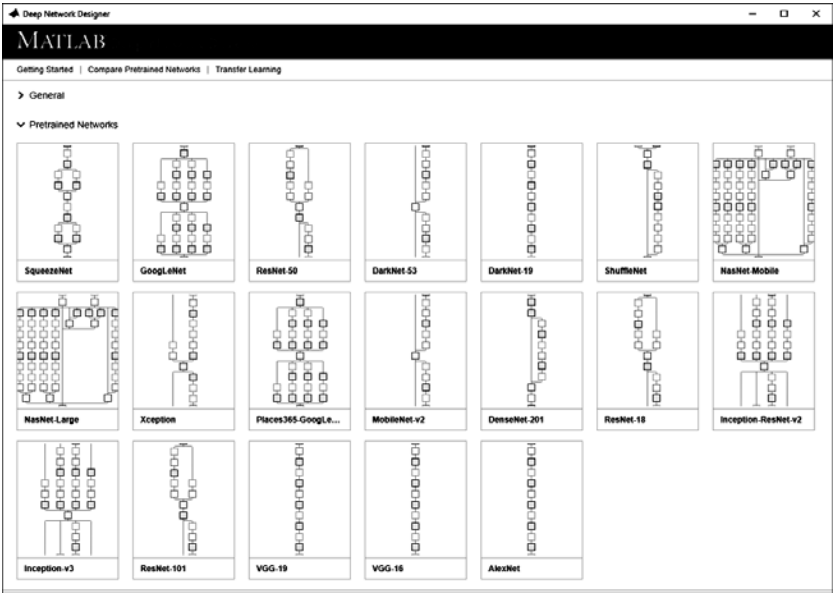


Рисунок 6.12 – Головна форма засобу DeepNetworkDesigner

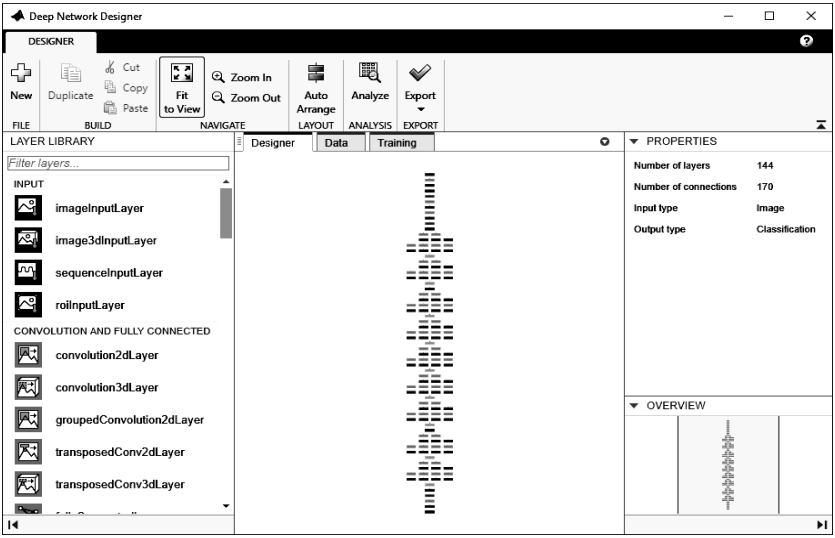


Рисунок 6.13 – Форма створення нейромереж засобу DeepNetworkDesigner

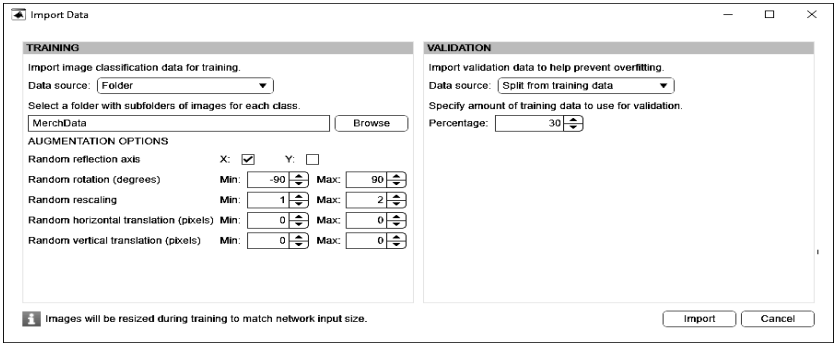


Рисунок 6.14 – Форма імпорту даних засобу DeepNetworkDesigner

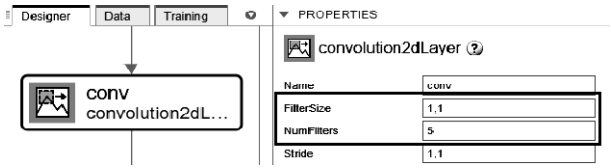


Рисунок 6.15 – Форма властивостей згорткового шару засобу DeepNetworkDesigner

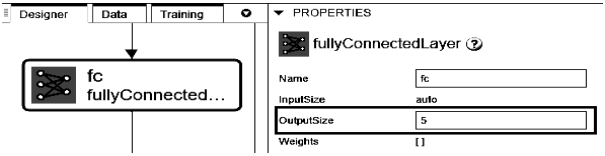


Рисунок 6.16 Форма властивостей повнозв'язного шару засобу DeepNetworkDesigner

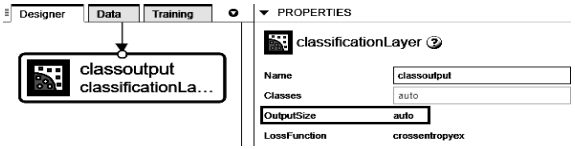


Рисунок 6.17 Форма властивостей вихідного шару засобу DeepNetworkDesigner

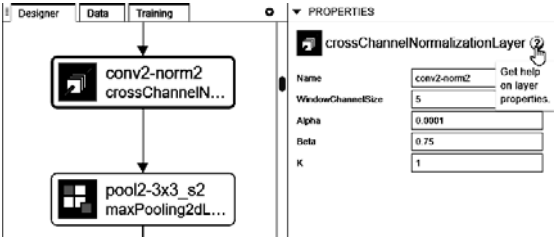


Рисунок 6.18 – Форма властивостей шару нормалізації

Також можна виділити тестові дані з навчальних даних. Обравши місце навчальних даних, вкажіть тестові дані та встановіть будь-які параметри збільшення, натисніть Import, щоб імпортувати набір даних.

Редагування попередньо навченої нейромережі для переданого навчання (передане навчання – це процес налагодження ваг вже попередньо навченої мережі для її настроювання на конкретну задачу за заданою навчальною вибіркою, який дозволяє швидше будувати прикладну нейромодель у порівнянні з навчанням мережі з випадковими значеннями початкових ваг) здійснюють таким чином.

Спочатку відкривають Deep Network Designer, завантажують попередньо навчену нейромережу, обираючи її на початковій сторінці Deep Network Designer (наприклад, можна обрати SqueezeNet). Після цього треба натиснути "Open" для завантаження мережі.

Для підготовки мережі до переданого навчання треба замінити останній шар, що навчається, та останній класифікуючий шар.

Якщо останній шар, що навчається, це двовимірний згортковий шар (наприклад, шар 'conv10' у SqueezeNet), то перетягніть новий convolutional2dLayer на робоче полотно, встановіть властивість NumFilters рівній новій кількість класів, а FilterSize встановіть як 1,1, видаліть останній convolutional2dLayer і замість цього підключіть новий шар.

Якщо останній шар, що навчається, це повнозв’язний шар (наприклад, у GoogLeNet), то перетягніть новий fullyConnectedLayer на робоче полотно та встановіть властивість OutputSize рівною новій кількості класів, видаліть останній fullyConnectedLayer та підключіть замість нього новий шар, після чого видаліть вихідний класифікуючий шар, потім перетягніть новий classificationLayer на полотно та підключіть його замість старого шару.

Щоб перевірити, чи готова мережа до навчання, на вкладці "Designer" натисніть "Analyze".

Для навчання мережі оберіть вкладку "Training".

Для отримання інформації про властивості шару на панелі "Designer" треба обрати шар для перегляду та редагування властивостей. Далі треба натиснути значок довідки поруч із назвою шару.

Навчання нейромережі на основі даних, імпортованих у Deep Network Designer, здійснюють таким чином. На вкладці "Training" треба обрати "Train". Якщо треба налаштувати параметри навчання, то обирають "Training Options".

Якщо необхідно навчати мережу на інших типах даних, то на вкладці "Designer" треба обрати "Export" для експорту початкової архітектури мережі. Після цього можливо програмно навчати мережу.

Експорт архітектури мережі, створеної в Deep Network Designer, у робочу область. Для експорту архітектури мережі з початковими вагами на вкладці "Designer" натисніть "Export". Для експорту архітектури мережі з навченими вагами на вкладці "Training" натисніть "Export".

Генерація тексту програми Matlab для створення архітектури мережі. Для відтворення шарів мережі на вкладці "Designer" оберіть "Export > Generate Code". Крім того, можливо відтворити мережу, включаючи будь-які параметри, що можуть навчатися, обравши "Export > Generate Code with Initial Parameters". Після генерації тексту програми можна виконати такі дії: відтворити шари мережі, створеної засобом Deep Network Designer (для цього треба запустити створений скрипт), навчати мережу (для цього треба запустити скрипт та подати шари до функції TrainNetwork), проаналізувати текст скрипту, щоб навчитися створювати та підключати шари програмно, змінити шари (для цього треба відредагувати текст скрипту), запустити скрипт та імпортувати мережу назад у Deep Network Designer для редагування.

Генерація тексту програми Matlab для навчання. Щоб відтворити імпорт даних та навчання, на вкладці «Training» оберіть Export > Generate Code for Training.

Основні функції для програмування глибоких мереж наведено у табл. 6.3.

Таблиця 6.3 – Основні функції для моделювання
глибоких нейромереж у пакеті Matlab

Назва функції	Призначення функції
1	2
trainingOptions	Параметри навчання нейромережі
trainNetwork	Навчання нейромережі
analyzeNetwork	Аналіз архітектури нейромережі
squeezenet	Створення згорткової нейромережі SqueezeNet
googlenet	Створення згорткової нейромережі GoogLeNet
inceptionv3	Згорткова нейромережа Inception-v3
densenet201	Згорткова нейромережа DenseNet-201
mobilenetv2	Згорткова нейромережа MobileNet-v2
resnet18	Згорткова нейромережа ResNet-18
resnet50	Згорткова нейромережа ResNet-50
resnet101	Згорткова нейромережа ResNet-101
xception	Згорткова нейромережа Xception
inceptionresnetv2	Попередньо навчена згорткова нейромережа Inception-ResNet-v2
nasnetlarge	Попередньо навчена згорткова нейромережа NASNet-Large
nasnetmobile	Попередньо навчена згорткова нейромережа NASNet-Mobile
shufflenet	Попередньо навчена згорткова нейромережа ShuffleNet
darknet19	Згорткова нейромережа DarkNet-19
darknet53	Згорткова нейромережа DarkNet-53
alexnet	Згорткова нейромережа AlexNet
vgg16	Згорткова нейромережа VGG-16
vgg19	Згорткова нейромережа VGG-19
imageInputLayer	Шар введення зображення
image3dInputLayer	Тривимірний шар введення зображення
convolution2dLayer	Двовимірний згортковий шар
convolution3dLayer	Тривимірний згортковий шар
groupedConvolution2dLayer	Двовимірний груповий згортковий шар
transposedConv2dLayer	Транспонований двовимірний згортковий шар
transposedConv3dLayer	Транспонований тривимірний згортковий шар
fullyConnectedLayer	Повністю зв’язаний шар
reluLayer	Шар ReLU
leakyReluLayer	Тікучий шар ReLU
clippedReluLayer	Обрізаний ректифікований шар ReLU
eluLayer	Шар ELU

Продовження табл. 6.3

1	2
tanhLayer	Шар гіперболічного тангенсу (tanh)
batchNormalizationLayer	Шар нормалізації пакетів
crossChannelNormalizationLayer	Шар нормалізації локальної канальної реакції
dropoutLayer	Шар випадання
crop2dLayer	Двовимірний збиральний шар
crop3dLayer	Тривимірний збиральний шар
averagePooling2dLayer	Двовимірний шар об'єднання усередненням
averagePooling3dLayer	Тривимірний шар об’єднання усередненням
globalAveragePooling2dLayer	Двовимірний глобальний шар об’єднання усередненням
globalAveragePooling3dLayer	Тривимірний глобальний шар об’єднання усередненням
globalMaxPooling2dLayer	Двовимірний глобальний шар об’єднання максимізацією
globalMaxPooling3dLayer	Тривимірний глобальний шар об’єднання максимізацією
maxPooling2dLayer	Двовимірний шар об’єднання максимізацією
maxPooling3dLayer	Тривимірний шар об’єднання максимізацією
maxUnpooling2dLayer	Двовимірний шар роз’єднання максимізацією
additionLayer	Шар додавання
concatenationLayer	Шар з’єднання
depthConcatenationLayer	Шар з’єднання в глибину
softmaxLayer	Шар Softmax
classificationLayer	Вихідний класифікуючий шар
regressionLayer	Вихідний регресійний шар
augmentedImageDatastore	Перетворення пакетів для збільшення даних зображень
imageDataAugmenter	Налаштування збільшення даних зображеннь
augment	Застосування ідентичних випадкових перетворень до множини зображень
layerGraph	Схема шарів мережі для глибокого навчання
plot	Побудова схеми шарів мережі
addLayers	Додавання шарів до схеми шарів
removeLayers	Видалення шарів зі схеми шарів
replaceLayer	Заміна шару у схемі шарів
connectLayers	З’єднання шарів у схемі шарів

Продовження табл. 6.3

1	2
disconnectLayers	Роз'єднання шарів у схемі шарів
DAGNetwork	Створення нейромережі Directed acyclic graph (DAG)
classify	Класифікація даних на основі навченої глибокої нейромережі
activations	Обчислення активацій шару глибокої нейромережі
predict	Передбачення виходів на основі навченої глибокої нейромережі
confusionchart	Створення діаграми матриці помилок для задач класифікації
sortClasses	Сортування класів діаграми матриці помилок

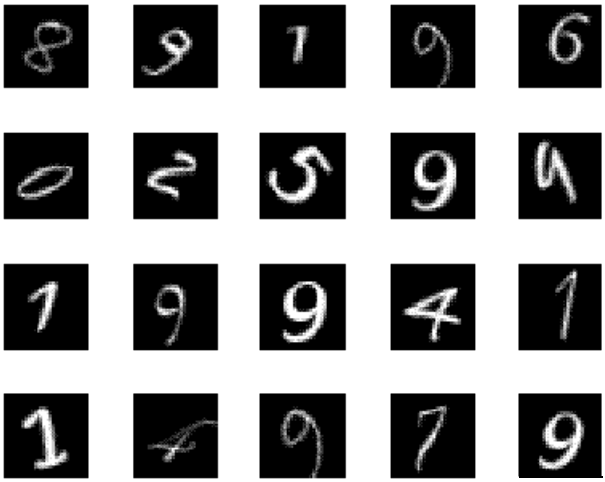
Розглянемо *створення простої згорткової нейронної мережі* з глибоким навчанням для класифікації (приклад TrainABasicConvolutionalNeuralNetworkForClassificationExample.m).

```
% Завантаження даних - вибірки зображень цифр
% як об'єкта ImageDatastore: imageDatastore – функція,
% що маркує зображення автоматично на основі імен папок
% та зберігає дані як об'єкт ImageDatastore, який дозволяє
% зберігати великі дані зображень, включаючи ті,
% що не уміщуються до пам'яті, та ефективно зчитує пакети
% зображень протягом навчання мережі.
digitDatasetPath = fullfile(matlabroot,'toolbox','nnet', ...
'nndemos', 'nndatasets','DigitDataset');
digitData = imageDatastore(digitDatasetPath, ...
        'IncludeSubfolders',true,'LabelSource','foldernames');

% Показ деяких зображень з об'єкту datastore.
figure;
perm = randperm(10000,20);
for i = 1:20
    subplot(4,5,i);
    imshow(digitData.Files{perm(i)});
end

% Обчислення кількості зображень у кожному класі
% CountLabel - таблиця, що містить мітки кластів та
% номери зображень у кожному класі.
CountLabel = digitData.countEachLabel;

% Перевірка та друк розміру першого зображення в об'єкті digitData.
img = readimage(digitData,1);
size(img)

% Визначення навчальної та тестової вибірок
% Дані розподіляються на навчальні та тестові так,
% щоб кожний клас у навчальній вибірці мав по 750 зображень,
% а решта - потрапляють у тестову вибірку.
% Функція splitEachLabel розбиває зображення хранилища digitData
```

```
% на два нових хранилища зображень trainDigitData та testDigitData.
trainingNumFiles = 750;
rng(1) % For reproducibility
[trainDigitData,testDigitData] = splitEachLabel(digitData, ...
trainingNumFiles,'randomize');
% Визначення архітектури згорткової нейромережі:
% Image Input Layer - вхідний шар мережі - визначається
% розміром вхідного зображення: висота, ширина, та
% розмір каналу (1 - тони сірого, 3 - кольоровий);
% Convolutional Layer - згортковий шар мережі: перший аргумент
% filterSize - висота та ширина фільтрів, які використовує функція
% навчання підчас перегляду зображень (у прикладі число 5 позначає,
% що розмір фільтра - [5,5]), другий аргумент - кількість фільтрів,
% що є кількістю нейронів, пов’язаних з цим регіоном виходу
% (визначає кількість карт ознак);
% ReLU Layer - шар вузлів з нелійнійною функцією активації;
% Max-Pooling Layer - шар, що зменшує розмірність даних.
% Він повертає максимальні значення прямокутних регіонів входів,
% заданих першим аргументом, розмір прямокутного регіону - [2,2],
% необов’язковий аргумент Stride визначає розмір кроку, з яким
% функція навчання проходить по зображенню;
% Fully Connected Layer - повнозв'язний шар, кількість вузлів
% на останньому повнозв'язному шарі має дорівнювати кількості
% розпізнаваних класів;
% Softmax Layer - повнозв'язний шар, що використовує
% функцію активації softmax;
% Classification Layer - останній шар мережі,
% що видає клас розпізнаваного екземпляра.
layers = [imageInputLayer([28 28 1])
          convolution2dLayer(5,20)
          reluLayer
          maxPooling2dLayer(2,'Stride',2)
          fullyConnectedLayer(10)
          softmaxLayer
          classificationLayer()];

% Визначення параметрів навчання: 'sgdm' - метод навчання
% нейромережі - стохастичний градієнтний спуск з моментом,
% 'MaxEpochs' - максимальна кількість повних
% циклів навчання (епох),15 - значення MaxEpochs,
% 'InitialLearnRate' - початкове значення кроку навчання,
% 0.0001 – значення InitialLearnRate.
options = trainingOptions('sgdm','MaxEpochs', ...
15,'InitialLearnRate',0.0001);

% Навчання мережі на основі заданих параметрів
% та навчальної вибірки даних.
convnet = trainNetwork(trainDigitData,layers,options);

%% Розпізнавання зображень тестової вибірки.
YTest = classify(convnet,testDigitData);
TTest = testDigitData.Labels;

% Обчислення точності моделі.
accuracy = sum(YTest == TTest)/numel(TTest)
```

Результати роботи програми

```
TrainABasicConvolutionalNeuralNetworkForClassificationExample
ans =
 28 28
Training on single CPU.
Initializing image normalization.
|=========================================================================|
|Epoch |Iteration|Time Elapsed|Mini-batch|Mini-batch|Base Learning|
|      |         |   (seconds)|    Loss  | Accuracy |     Rate    |
|=========================================================================|
|    1 |      1  |      3.50  |  3.0845  |  13.28%  |  1.00e-04   |
|    1 |     50  |     61.74  |  1.0945  |  65.63%  |  1.00e-04   |
|    2 |    100  |    105.03  |  0.7281  |  74.22%  |  1.00e-04   |
|    3 |    150  |    143.27  |  0.4736  |  83.59%  |  1.00e-04   |
|    4 |    200  |    182.55  |  0.3092  |  91.41%  |  1.00e-04   |
|    5 |    250  |    244.61  |  0.2325  |  92.97%  |  1.00e-04   |
|    6 |    300  |    289.25  |  0.1540  |  97.66%  |  1.00e-04   |
|    7 |    350  |    327.57  |  0.1314  |  97.66%  |  1.00e-04   |
|    7 |    400  |    365.52  |  0.0943  |  96.09%  |  1.00e-04   |
|    8 |    450  |    404.55  |  0.0665  |  99.22%  |  1.00e-04   |
|    9 |    500  |    442.89  |  0.0460  |  99.22%  |  1.00e-04   |
|   10 |    550  |    482.56  |  0.0545  | 100.00%  |  1.00e-04   |
|   11 |    600  |    520.64  |  0.0658  |  99.22%  |  1.00e-04   |
|   12 |    650  |    559.68  |  0.0338  | 100.00%  |  1.00e-04   |
|   13 |    700  |    601.57  |  0.0342  | 100.00%  |  1.00e-04   |
|   13 |    750  |    639.85  |  0.0370  |  99.22%  |  1.00e-04   |
|   14 |    800  |    678.92  |  0.0264  | 100.00%  |  1.00e-04   |
|   15 |    850  |    717.38  |  0.0181  | 100.00%  |  1.00e-04   |
|   15 |    870  |    732.74  |  0.0235  | 100.00%  |  1.00e-04   |
|=========================================================================|
accuracy = 0.9808
```



Опис результатів роботи програми.

Після запуску скрипту TrainABasicConvolutionalNeuralNetwork-ForClassificationExample програма видає розмір зображення.

В окремому вікні видається зображення зі зразками вхідних образів.

Далі зазначається, що тренування виконувалося на одному процесорі. За наявності більшої кількості процесорів або графічної карти програма автоматично обере потрібну кількість процесорів, яку виведе на екран.

Після чого видається підтвердження того, що йде процес нормалізації зображень.

Далі послідовно видається підчас навчання таблиця, що характеризує процес навчання: Epoch – номер виконаної епохи, Iteration — номер поточної ітерації, Time Elapsed – обсяг витраченого часу, Mini-batch loss – втрати для мініпакету (підвибірки), точність мініпакету, базовий крок навчання (коригувальний приріст)

Після таблиці видається точність навчання для тестової вибірки.

Налаштування попередньо навченої згорткової нейромережі. При необхідності побудувати нейромодель за обмеженою вибіркою спостережень, наявною у користувача, може виявитися більш ефективною технологія transfer learning – передачі навчання, коли користувач отримує попередньо навчену на великому масиві даних мережу, а потім донавчає її на подібних спостереженнях, які він має.

```
% Завантаження попередньо навченої мережі у змінній net з файлу
% LettersClassificationNet.mat, що навчена на великій колекції
% зображень літер у тонах сірого розміром 28x28 пікселів.
% Мережа розподіляє літери на три класи: 'A', 'B' та 'C'.
load(fullfile(matlabroot,'examples','nnet',...
'LettersClassificationNet.mat'))

% Перевірка параметрів архітектури мережі, що містяться
% у властивості Layers об'єкта net.
net.Layers

% Завантаження навчальної вибірки даних зображень цифр як
% об’єкту ImageDatastore. Функція imageDatastore автоматично
% позначає зображення мітками класів, грунтуючись на іменах
% папок, та зберігає дані як об’єкт ImageDatastore, який
% дозволяє зберігати великі дані зображень, включаючи ті,
% що не уміщуються у пам’яті. Цей об’єкт також дозволяє
% ефективно зчитувати пакети зображень підчас навчання.
% Хранилище даних містить 10000 штучно синтезованих зображень
% цифр 0-9. Зображення згенеровані застосуванням випадкового
```

```
% перетворення до зображень цифр, створених з використанням
% різних гарнітур шрифтів. Кожне зображення має розмір
% 28x28 пікселів.
digitDatasetPath = fullfile(matlabroot,'toolbox','nnet',...
'nndemos', 'nndatasets','DigitDataset');
digitData = imageDatastore(digitDatasetPath, ...
'IncludeSubfolders',true,'LabelSource','foldernames');

% Розбиття вибірки даних на навчальну та тестову вибірки
% таким чином, щоб кожне хранилище даних мало 50% зображень
% кожного класу. splitEachLabel розбиває файли зображень
% в digitData на два нових хранилища даних: trainDigitData
% та testDigitData.
[trainDigitData,testDigitData] = splitEachLabel(digitData,...
0.5,'randomize');

% Показ 20 випадкових зразків зображень навчальної вибірки
numImages = numel(trainDigitData.Files);
idx = randperm(numImages,20);
for i = 1:20
    subplot(4,5,i)
    I = readimage(trainDigitData, idx(i));
    imshow(I)
end

% Передача шарів до цільової мережі.
% Оскільки останні три шари попередньо навченої мережі net
% навчені для літер, то їх треба замінити та навчити для цифр.
% Тому здійснюється витяг усіх шарів, окрім трьох останніх,
% з попередньо навченої мережі
layersTransfer = net.Layers(1:end-3);

% Оскільки дані цифр мають 10 класів, то додамо новий
% повнозв'язний шар для 10 класів, шар softmax та
% вихідний шар класифікації.
% Для підвищення швидкості навчання тільки для нових шарів
% збільшимо кроки навчання для ваг та порогів повнозв'язного шару
numClasses = numel(categories(trainDigitData.Labels))
layers = [layersTransfer fullyConnectedLayer(numClasses, ...
'WeightLearnRateFactor',20,'BiasLearnRateFactor', ...
20) softmaxLayer classificationLayer];

% Якщо вхідні зображення відрізняються за розміром від
% вхідного шару зображень, тоді треба змінити розмір даних
% зображень. У цьому прикладі розміри співпадають.
% Визначення параметрів процесу навчання.
% Для передачі навчання треба зберегти ознаки з попередніх
% шарів попередньо навченої мережі (передача ваг шарів).
% Для цього встановимо InitialLearnRate як мале значення.
% Це мале значення кроку навчання буде уповільнювати
% навчання на переданих шарах.
% На попередньому кроці для повнозв'язного шару встановлені
% великі кроки навчання для пришвидшення навчання нових
% останніх шарів.
% Така комбінация призваеде до швидкого навчання тільки
% нових шарів, фіксуючи інші шари. Передача навчання
```

```
% не потребує багато циклів тренування (епох).
% Для пришвидшення навчання доцільно зменшити значення
% параметру 'MaxEpochs' при виклиці trainingOptions.
optionsTransfer = trainingOptions('sgdm', ...
    'MaxEpochs',5, ...
    'InitialLearnRate',0.0001);

% Донавчання мережі, що складається з переданих та нових шарів
netTransfer = trainNetwork(trainDigitData,layers,optionsTransfer);
% Обчислення точності класифікації - долі правильно
% розпізнаних зразків тестової вібирки
YPred = classify(netTransfer,testDigitData);
YTest = testDigitData.Labels;
accuracy = sum(YPred==YTest)/numel(YTest)

% Показ прикладів тестових зображень з їхніми мітками класів,
% розпізнаними мережею.
idx = 501:500:5000;
figure
for i = 1:numel(idx)
    subplot(3,3,i)
    I = readimage(testDigitData, idx(i));
    label = char(YTest(idx(i)));
    imshow(I)
    title(label)
end
```

Результати роботи програми.

```
ans =
 7x1 Layer array with layers:
  1 'imageinput' Image Input
           28x28x1 images with 'zerocenter' normalization
  2 'conv' Convolution
           20 5x5x1 convolutions with stride [1 1] and padding [0 0]
  3 'relu'   ReLU     ReLU
  4 'maxpool' Max Pooling
           2x2 max pooling with stride [2 2] and padding [0 0]
  5 'fc'   Fully Connected
           3 fully connected layer
  6 'softmax'  Softmax     softmax
  7 'classoutput' Classification Output
           crossentropyex with 'A', 'B', and 1 other classes
numClasses = 10
Training on single CPU.
Initializing image normalization.
|=========================================================================|
|Epoch|Iteration|Time Elapsed|Mini-batch|Mini-batch| Base Learning|
|     |         |   seconds) |   Loss   | Accuracy |     Rate     |
|=========================================================================|
|   1 |       1 |     1.57   |    9.4193|   11.72% |   1.00e-04   |
|   2 |      50 |    40.01   |    3.6911|   64.06% |   1.00e-04   |
|   3 |     100 |    78.50   |    1.7290|   82.81% |   1.00e-04   |
|   4 |     150 |    117.13  |    1.2480|   88.28% |   1.00e-04   |
|   5 |     195 |    151.77  |    0.6499|   89.84% |   1.00e-04   |
|=========================================================================|
accuracy =    0.8946
```

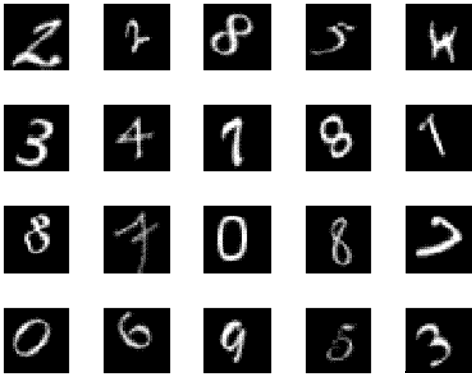


Figure 1

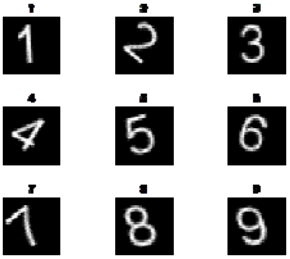


Figure 2

Опис результатів роботи програми. Після запуску програма видає значення параметрів архітектури мережі. Мережа містить сім шарів. Шари 2-4 знаходять ознаки. Шари 5-7 перетворюють ознаки у класи літер. 20 виходів (каналів) шару згорток відповідають вивченим ознакам. Останні три шари попередньо навченої мережі net налаштовуються за даними літер. Шляхом заміни останніх трьох шарів попередньо навченої мережі можливо навчити мережу класифікувати цифри.

Далі видається кількість класів numClasses = 10

Після чого повідомляється, що навчання здійснюється на одному процесорі.

Далі зазначається, що виконується нормалізація зображень.

Після чого у процесі навчання послідовно видається таблиця з характеристиками процесу навчання: Epoch – номер виконаної епохи, Iteration – номер поточної ітерації, Time Elapsed – обсяг витраченого часу, Mini-batch – втрати для мініпакету (підвибірки), точність мініпакету, базовий крок навчання (коригувальний приріст)

Після таблиці видається точність навчання для тестової вибірки.

На окремих рисунках видаються 20 випадкових зображень з навчальної вибірки (figure 1) та 9 забражень тестової вибірки зі співставленими ним мітками класів мережею (figure 2).

Навчання згорткової нейронної мережі регресії. Розглянемо приклад передбачення кутів повороту рукописних цифр за допомогою згорткової нейромережі.

```
% Завантаження навчальних даних як чотиривимірного масиву.
% Мережа навчатиметься на колекції штучно синтезованих
% рукописних символах (5000 зображень цифр з відповідними
% углами повороту).
[trainImages,~,trainAngles] = digitTrain4DArrayData;

% Демонстрація 20 випадкових зразків навчальних цифр.
numTrainImages = size(trainImages,4);
figure
idx = randperm(numTrainImages,20);
for i = 1:numel(idx)
    subplot(4,5,i)
    imshow(trainImages(:,:,:,idx(i)))
    drawnow
end

% Створення шарів мережі
% Перший шар визначає розмір та тип вхідних даних.
% Вхідні зображення мають розмір 28x28 точок та
% 1 канал у тонах сірого.
% Середні шари визначають ядро архітектури мережі.
% Створюється двовимірний згортковий шар з 25 фільтрами
% розміром 12, за яким йде шар ReLU.
% Останні шари визначають розмір та тип вихідних данихю.
% Для регресії повнозв'язаний шар має передувати
% регресійному шару, що розміщується у кінці мережі.
% Створюються повнозв'язний шар розміру 1 та регресійний шар.
% Далі усі шари поєднуються в масиві Layer.
layers = [ ...
    imageInputLayer([28 28 1])
    convolution2dLayer(12,25)
    reluLayer
    fullyConnectedLayer(1)
    regressionLayer];

% Навчання мережі
% Визначення параметрів навчання мережі.
% Установка начального значення кроку навчання у 0.001.
```

```
% Для зменшення часу навчання можна зменшити значення 'MaxEpochs'.
options = trainingOptions('sgdm','InitialLearnRate',0.001, ...
'MaxEpochs',15);

% Створення нейромережі за допомогою функції trainNetwork.
net = trainNetwork(trainImages,trainAngles,layers,options)

% Перевірка параметрів архітектури мережі,
% що містяться у властивості Layers об'єкту net.
net.Layers

% Перевірка якості мережі шляхом обчислення точності
% передбачення для тестових даних.
% Завантаження тестової множини цифр.
[testImages,~,testAngles] = digitTest4DArrayData;

% Функція predict передбачає кути поворотів тестових зображень.
predictedTestAngles = predict(net,testImages);

% Обчислення якості моделі.
% Обчислення помилки передбачення між передбаченими
% та дійсними кутами поворотів.
predictionError = testAngles - predictedTestAngles;

% Обчислення кількості передбачень у межах прийнятної
% помилки відносно дійсних кутів. Установка порогу
% у 10 градусів. Обчислення відсотку передбачень з цим порогом.
thr = 10;
numCorrect = sum(abs(predictionError) < thr);
numTestImages = size(testImages,4);
accuracy = numCorrect/numTestImages

% Обчислюємо середньоквадратичну помилку (RMSE) для вимірювання
% різниці між передбаченим та справжнім кутами повороту.
squares = predictionError.^2;
rmse = sqrt(mean(squares))

% Обчислення залишків.
residuals = testAngles - predictedTestAngles;

% Функція boxplot вимагає матрицю, де кожний стовпець відповідає
% залишкам кожного класу цифр.
% Тестові дані містять зображення цифр 0-9
% по 500 зразків кожного класу.
% Функція reshape групує залишки за класами цифр.
% Кожний стовпець residualMatrix відповідає залишкам
% для відповідної цифри.
residualMatrix = reshape(residuals,500,10);

% Зображення залишків для кожного класу цифр
% у вигляді діаграм "шухляда з вусами"
figure
boxplot(residualMatrix, ...
'Labels',{'0','1','2','3','4','5','6','7','8','9'})
xlabel('Digit Class')
ylabel('Degrees Error')
```

```
title('Residuals')

% Класи цифр з найвищою точністю мають
% середні, близькі до нуля, та незначні дисперсії.
% Корекція поворотів цифр
idx = randperm(numTestImages,49);
for i = 1:numel(idx)
    image = testImages(:,:,:,idx(i));
    predictedAngle = predictedTestAngles(idx(i));
    imagesRotated(:,:,:,i) = imro-
tate(image,predictedAngle,'bicubic','crop');
end

% Демонстрація оригінальних зображень цифр та
% їхніх відкоректованих поворотів.
figure
subplot(1,2,1)
montage(testImages(:,:,:,idx))
title('Original')
subplot(1,2,2)
montage(imagesRotated)
title('Corrected')
```

Результати роботи програми.

```
>> TrainAConvolutionalNeuralNetworkForRegressionExample
Training on single CPU.
Initializing image normalization.
|=========================================================================|
| Epoch |Iteration|Time Elapsed|Mini-batch|Mini-batch|Base Learning|
|       |         |  (seconds) |   Loss   |   RMSE   |     Rate     |
|=========================================================================|
|     1 |       1 |       3.71  | 392.5880 |    28.02 |     0.0010   |
|     2 |      50 |      96.96  |  96.0224 |    13.86 |     0.0010   |
|     3 |     100 |     185.28  |  55.5496 |    10.54 |     0.0010   |
|     4 |     150 |     270.46  |  64.4647 |    11.35 |     0.0010   |
|     6 |     200 |     352.72  |  36.2445 |     8.51 |     0.0010   |
|     7 |     250 |     433.93  |  53.0401 |    10.30 |     0.0010   |
|     8 |     300 |     513.46  |  35.7745 |     8.46 |     0.0010   |
|     9 |     350 |     592.11  |  34.9632 |     8.36 |     0.0010   |
|    11 |     400 |     670.20  |  24.9389 |     7.06 |     0.0010   |
|    12 |     450 |     748.11  |  33.0234 |     8.13 |     0.0010   |
|    13 |     500 |     830.36  |  27.1019 |     7.36 |     0.0010   |
|    15 |     550 |     915.09  |  18.1279 |     6.02 |     0.0010   |
|    15 |     585 |     968.85  |  18.9767 |     6.16 |     0.0010   |
|=========================================================================|
net =
    SeriesNetwork with properties:
    Layers: [5×1 nnet.cnn.layer.Layer]
ans =
 5x1 Layer array with layers:
1 'imageinput' Image Input
         28x28x1 images with 'zerocenter' normalization
2 'conv' Convolution
         25 12x12x1 convolutions with stride [1 1] and padding [0 0]
3 'relu' ReLU ReLU
```

```
4 'fc'   Fully Connected 1 fully connected layer
5 'regressionoutput' Regression Output
         mean-squared-error with response 'Response'
accuracy = 0.8488
rmse =
  single
  7.3413
```

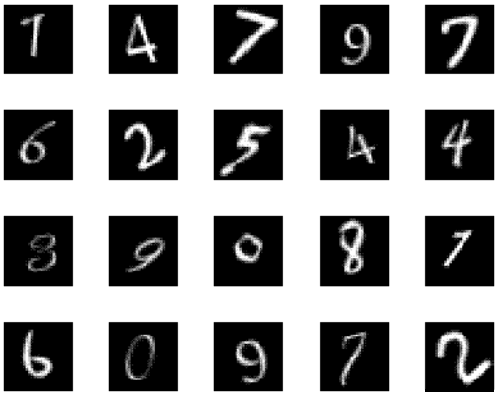


figure 1

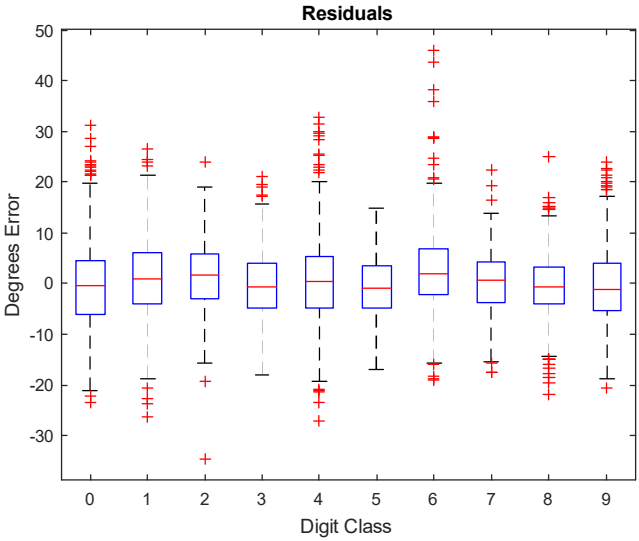


figure 2

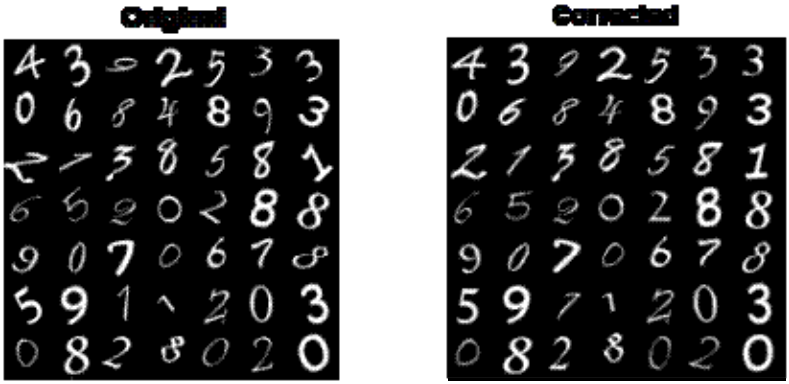


figure 3

Опис результатів роботи програми. Після запуску програми вона зазначає, що навчання відбувається на одному процесорі.

Далі повідомляється, що виконується нормалізація зображень.

Після чого у процесі навчання послідовно видається таблиця з характеристиками процесу навчання: Epoch – номер виконаної епохи, Iteration – номер поточної ітерації, Time Elapsed – обсяг витраченого часу, Mini-batch loss – втрати для мініпакету (підвибірки), Mini-batch RMSE – середньоквадратична помилка для мініпакету, Base Learning Rate – базовий крок навчання (коригувальний приріст).

Після таблиці видається опис мережі об’єкту net. Зазначається, що це – серійна мережа, що складається з 5x1 шарів:

1 'imageinput' – вхідне зображення (28x28x1 з нормалізацією 'zerocenter');

2 'conv' – згортковий (25 12x12x1 згорток з кроком [1 1] та заповненням [0 0]);

3 'relu' – ReLU;

4 'fc' –повнозв’язний шар;

5 'regressionoutput' – регресійний вихід (середньоквадратична помилка з відгуком 'Response').

Після чого видаються точність accuracy = 0.8488 та середньоквадратична помилка rmse = 7.3413

На окремих рисунках видаються приклади випадкових зображень з навчальної множини (figure 1), діаграма залишків за класами цифр

(figure 2) та порівняння випадкових зображень тестової множини та їхніх корекцій на передбачену величину кута повороту (figure 3).

Класифікація послідовностей даних на основі LSTM мереж. Для класифікації послідовностей даних у пакеті Matlab можливо використовувати мережі довгої короткочасної пам'яті (Long Short-Term Memory – LSTM).

Розглянемо приклад навчання LSTM-мережі розпізнаванню мовця за двома японськими голосними, вимовленими послідовно.

```
% Завантаження навчальних даних – масиву звуків японської мови
% X, що містить 270 послідовностей розмірності 12 різної довжини.
% Y – це категоріальний вектор з мітками "1","2",...,"9",
% що відповідають 9 особам, яки вимовляли звуки.
load JapaneseVowelsTrain

% Перегляд п'яти перших спостережень.
% Записи в X – це матриці з 12 рядків (один рядок на кожну ознаку)
% та змінною кількістю стовпців (один стовпець на кожний крок часу).
X(1:5)

% Візуалізація перших часових рядів
% (кожен вбудований графік відповідає ознаці)
figure
for i = 1:12
    subplot(12,1,13-i)
    plot(X{1}(i,:));
    ylabel(i)
    xticklabels('')
    yticklabels('')
    box off
end
title("Training Observation 1")
subplot(12,1,12)
xticklabels('auto')
xlabel("Time Step")

% Отримання довжини посідовності для кожного спостереження.
numObservations = numel(X);
for i=1:numObservations
    sequence = X{i};
    sequenceLengths(i) = size(sequence,2);
end

% Упорядковування даних за довжиною послідовностей.
% Під час тренувань навчальні дані розбиваються на міні-партії
% та набиває або обрізає послідовності так, щоб вони мали однакову
% довжину. Завеликі набивка або видалення даних можуть негативно
% вплинути на продуктивність мережі. Для запобігання цьому можна
% відсортувати навчальні дані за довжиною послідовності та обрати
% розмір міні-пакету так, щоб послідовності у міні-пакеті мали
% однакову довжину.
[sequenceLengths,idx] = sort(sequenceLengths);
```

```
X = X(idx);
Y = Y(idx);

% Перегляд упорядкованих довжин послідовностей
% на стовпчиковій діаграмі.
figure
bar(sequenceLengths)
ylim([0 30])
xlabel("Sequence")
ylabel("Length")
title("Sequence Lengths")

% Встановлення обмежень міні-пакету
% Розмір міні-пакету 27 поділяє навчальні дані рівномірно
% та зменшує кількість набивок в міні-пакетах.
miniBatchSize = 27;
miniBatchLocations = miniBatchSize+1:miniBatchSize:numObservations;
XLocations = repmat(miniBatchLocations,[2 1]);
YLocations = repmat([0;30],[1 9]);

% Зображення обмежень міні-пакетів з довжинами послідовностей.
% Зображення показує як послідовності розподіляються
% на міні-пакети.
hold on
line(XLocations,YLocations, 'Color','r', 'LineStyle','--')

% Визначення архітектури LSTM-мережі: розмірності вхідних даних
% inputSize, розмірності вихідних даних outputSize, режиму видачі
% (outputMode) – останній елемент послідовності 'last', кількості
% класів numClasses.
inputSize = 12;
outputSize = 100;
outputMode = 'last';
numClasses = 9;

% Визначення параметрів навчання: максимальної кількості
% епох навчання maxEpochs, розміру міні-пакету miniBatchSize
% та того, що не треба перетасовувати дані.
maxEpochs = 150;
miniBatchSize = 27;
shuffle = 'never';
options = trainingOptions('sgdm', 'MaxEpochs',maxEpochs, ...
    'MiniBatchSize',miniBatchSize,'Shuffle', shuffle);

% Навчання LSTM-мережі.
net = trainNetwork(X,Y,layers,options);

% Перевірка навченої мережі
% Завантаження тестової вибірки та класифікація послідовностей
% за мовцями.XTest – масив, що містить 370 послідовностей
% розмірністю 12 різної довжини. YTest – категоріальний вектор
% міток "1","2",..."9", що відповідають дев’яти мовцям.
load JapaneseVowelsTest
XTest(1:3)

% Візуалізуємо перші часові ряди на графіку.
```

```
% Кожен вбудований графік відповідає ознаці.
figure
for i = 1:12
    subplot(12,1,13-i)
    plot(XTest{1}(i,:))
    ylabel(i)
    xticklabels('')
    yticklabels('')
    box off
end
title("Test Observation 1")
subplot(12,1,12)
xticklabels('auto')
xlabel("Time Step")

% Мережа net навчена з використанням міні-пакетів
% однакової довжини. Пересвідчимося, що тестові дані мають
% таку ж довжину та організовані подібним чином.
% Упорядкуємо тестові дані за довжинлю послідовності.
numObservationsTest = numel(XTest);
for i=1:numObservationsTest
    sequence = XTest{i};
    sequenceLengthsTest(i) = size(sequence,2);
end
[sequenceLengthsTest,idx] = sort(sequenceLengthsTest);
XTest = XTest(idx);
YTest = YTest(idx);

% Класифікація тестових даних.
% Для зменшення кількості набивок, уведених процесом
% класифікації встановлюється розмір міні-пакету 27.

miniBatchSize = 27;
YPred = classify(net,XTest, 'MiniBatchSize',miniBatchSize);

% Обчислення точності класифікації
acc = sum(YPred == YTest)./numel(YTest)
```

Результати роботи програми.

```
ans = 5×1 cell array
    {12×20 double}
    {12×26 double}
    {12×22 double}
    {12×20 double}
    {12×21 double}

layers =
5x1 Layer array with layers:
1 ''   Sequence Input          Sequence input with 12 dimensions
2 ''   LSTM                   LSTM with 100 hidden units
3 ''   Fully Connected         9 fully connected layer
4 ''   Softmax                softmax
5 ''   Classification Output   crossentropyex

Training on single GPU.
```

```
|=========================================================================|
|Epoch|Iteration|Time Elapsed|Mini-batch|Mini-batch  |Base Learning|
|     |         |  (seconds) |   Loss   | Accuracy   |     Rate    |
|=========================================================================|
|   1 |    1    |    0.37    |  2.1973  |  0.00%     |  0.0100     |
|   5 |   50    |    1.36    |  2.2147  |  0.00%     |  0.0100     |
|  10 |  100    |    2.12    |  2.2042  |  0.00%     |  0.0100     |
|  15 |  150    |    2.89    |  2.1926  |  0.00%     |  0.0100     |
|  20 |  200    |    3.69    |  2.1652  |  14.81%    |  0.0100     |
|  25 |  250    |    4.46    |  2.0326  |  7.41%     |  0.0100     |
|  30 |  300    |    5.20    |  1.5640  |  3.70%     |  0.0100     |
|  35 |  350    |    5.96    |  1.1499  |  55.56%    |  0.0100     |
|  40 |  400    |    6.70    |  0.8349  |  62.96%    |  0.0100     |
|  45 |  450    |    7.44    |  0.7832  |  70.37%    |  0.0100     |
|  50 |  500    |    8.19    |  0.6772  |  74.07%    |  0.0100     |
|  55 |  550    |    8.93    |  0.5303  |  77.78%    |  0.0100     |
|  60 |  600    |    9.66    |  0.3584  |  88.89%    |  0.0100     |
|  65 |  650    |   10.44    |  0.2077  |  96.30%    |  0.0100     |
|  70 |  700    |   11.18    |  0.2477  |  88.89%    |  0.0100     |
|  75 |  750    |   11.99    |  0.0967  |  100.00%   |  0.0100     |
|  80 |  800    |   12.77    |  0.0551  |  100.00%   |  0.0100     |
|  85 |  850    |   13.56    |  0.1457  |  96.30%    |  0.0100     |
|  90 |  900    |   14.36    |  0.0581  |  100.00%   |  0.0100     |
|  95 |  950    |   15.14    |  0.0332  |  100.00%   |  0.0100     |
| 100 |  1000   |   15.89    |  0.0272  |  100.00%   |  0.0100     |
| 105 |  1050   |   16.64    |  0.0255  |  100.00%   |  0.0100     |
| 110 |  1100   |   17.42    |  0.0163  |  100.00%   |  0.0100     |
| 115 |  1150   |   18.18    |  0.0133  |  100.00%   |  0.0100     |
| 120 |  1200   |   18.94    |  0.0113  |  100.00%   |  0.0100     |
| 125 |  1250   |   19.70    |  0.0100  |  100.00%   |  0.0100     |
| 130 |  1300   |   20.48    |  0.0126  |  100.00%   |  0.0100     |
| 135 |  1350   |   21.32    |  0.1965  |  88.89%    |  0.0100     |
| 140 |  1400   |   22.12    |  0.0172  |  100.00%   |  0.0100     |
| 145 |  1450   |   22.91    |  0.0101  |  100.00%   |  0.0100     |
| 150 |  1500   |   23.69    |  0.0088  |  100.00%   |  0.0100     |
|=========================================================================|
ans = 3×1 cell array
    {12×19 double}
    {12×17 double}
    {12×19 double}
acc = 0.9432
```

Опис результатів роботи програми. Після запуску програми видається опис масиву з перших трьох спостережень та зображення графіків 12 ознак першого екземпляра навчальної вибірки (figure 1). Після цього на стовпчиковій діаграмі зображуються упорядковані довжини послідовностей (figure 2).

Далі після створення архітектури нейромережі видається її опис. Зазначається, що мережа складається з 5 шарів: 1 – вхідна послідовність (з 12 ознаками), 2 – LSTM зі 100 прихованими вузлами, 3 – повнозв'язний шар, 4 – шар Softmax, 5 – вихідний шар.

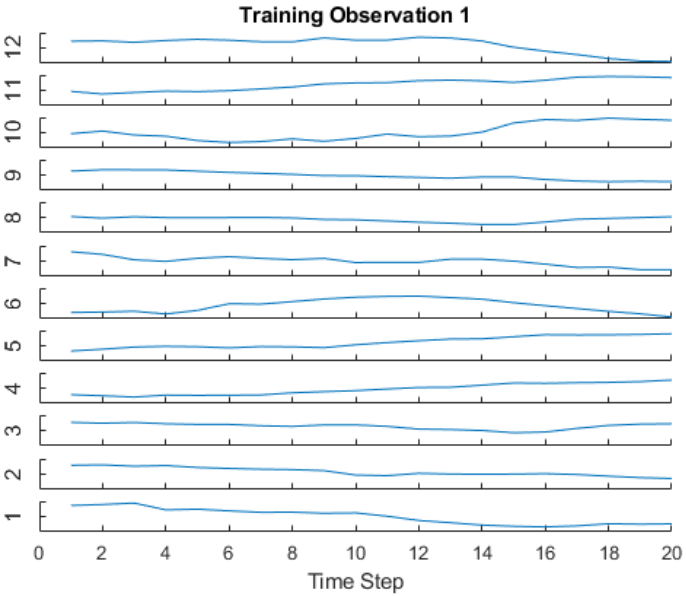


figure 1

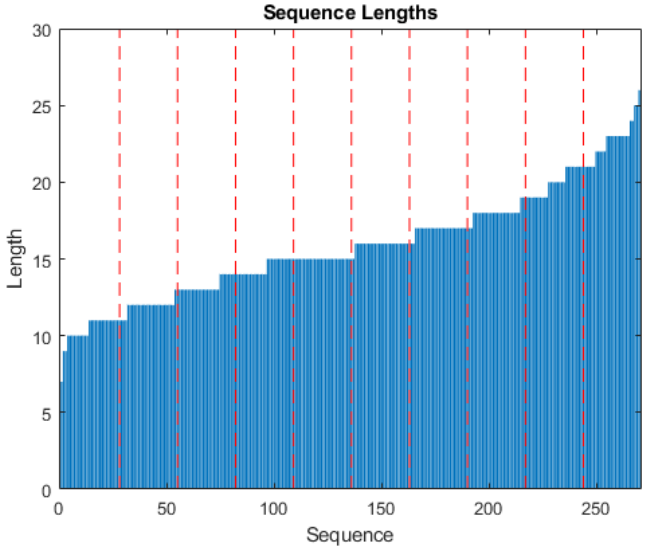


figure 2