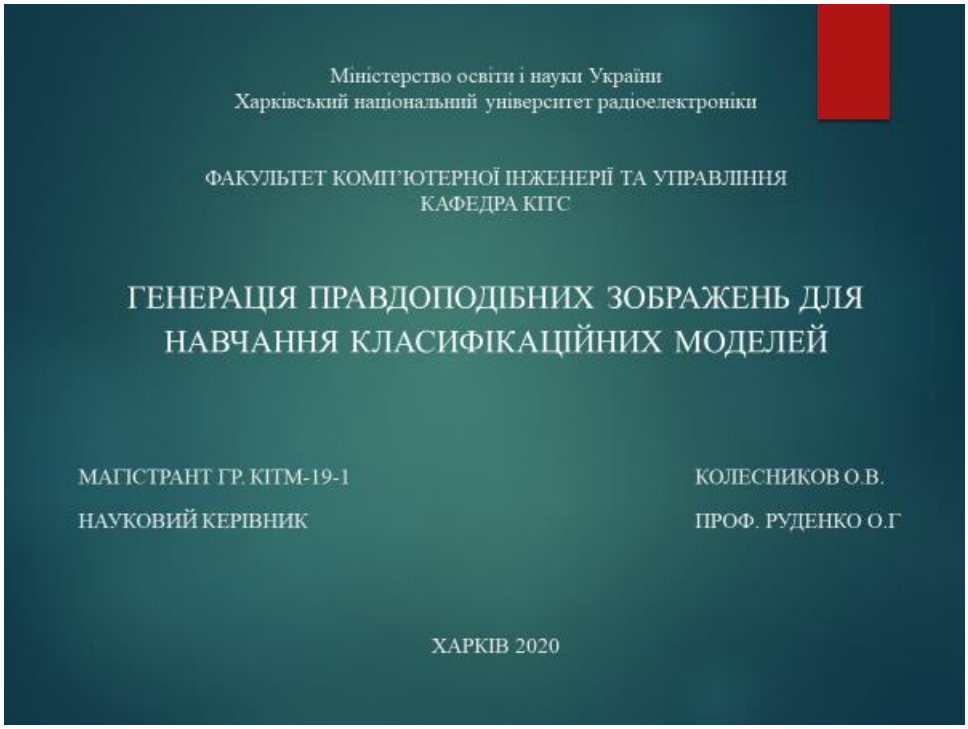
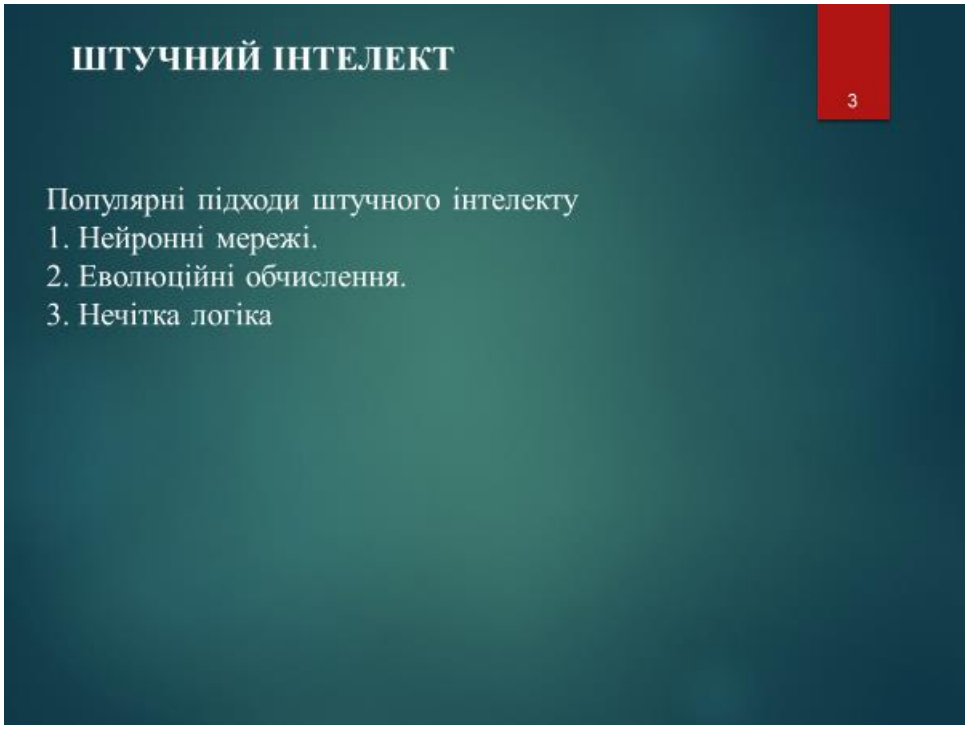
ДОДАТОК А

Графічний матеріал атестаційної роботи


Міністерство освіти і науки України
Харківський національний університет радіоелектроніки
ФАКУЛЬТЕТ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА УПРАВЛІННЯ
КАФЕДРА КІТС
ГЕНЕРАЦІЯ ПРАВДОПОДІБНИХ ЗОБРАЖЕНЬ ДЛЯ НАВЧАННЯ КЛАСИФІКАЦІЙНИХ МОДЕЛЕЙ
МАГІСТРАНТ ГР. КІТМ-19-1
КОЛЕСНИКОВ О.В.
НАУКОВИЙ КЕРІВНИК
ПРОФ. РУДЕНКО О.Г
ХАРКІВ 2020


ШТУЧНИЙ ІНТЕЛЕКТ
3
Популярні підходи штучного інтелекту
1. Нейронні мережі.
2. Еволюційні обчислення.
3. Нечітка логіка

МЕТА РОБОТИ

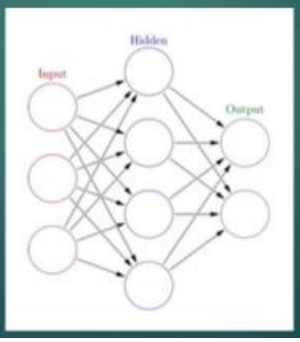
2

1. Розглянути, що таке штучний інтелект, нейронні мережі, які бувають види нейронних мереж.
2. Визначити методи та варіанти навчання класифікаційних моделей.
3. Створення програми для навчання моделі.
4. Проаналізувати отримані результати

НЕЙРОННІ МЕРЕЖІ

4

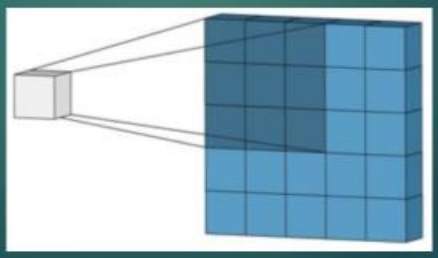
Цей напрямок стабільно тримається на першому місці. Триває вдосконалення алгоритмів навчання і класифікації в масштабі реального часу, обробки природних мов, розпізнавання зображень, мови, сигналів, а також 25 створення моделей інтелектуального інтерфейсу, підлаштовуватися під користувача. Серед основних прикладних завдань, що вирішуються за допомогою нейронних мереж, - фінансове прогнозування, розкопка даних, діагностика систем, контроль за діяльністю мереж, шифрування даних. В останні роки йде посилений пошук ефективних методів синхронізації роботи нейронних мереж на паралельних пристроях.



ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ

5

Згорткові нейронні мережі – це архітектура штучних нейронних мереж, запропонована Я. Лекуном у 1988 році і націлена на ефективне з боку потужностей та точності розпізнавання зображень, входить до складу технологій глибокого навчання. Використовує деякі особливості зорової кори, в якій були відкриті так звані прості клітини, що реагують на прямі лінії під різними кутами, і складні клітини, реакція яких пов'язана з активацією певного набору простих клітин. Таким чином, ідея згорткових нейронних мереж полягає в чергуванні двох основних шарів convolution layers і subsampling layers. Структура цієї мережі односпрямована, принципово багатошарова. Для навчання використовуються стандартні методи, такі як метод зворотного поширення помилки.

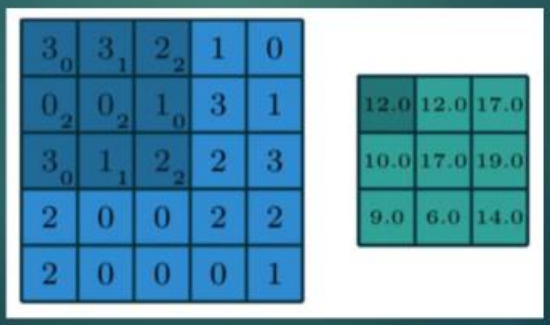


ОПЕРАЦІЯ ЗГОРТКИ

6

Архітектура отримала свою назву через наявність операції згортки, суть якої в тому, що кожен фрагмент зображення множиться на матрицю (ядро) згортки поелементно, а результат підсумовується і записується в аналогічну позицію вихідного зображення.

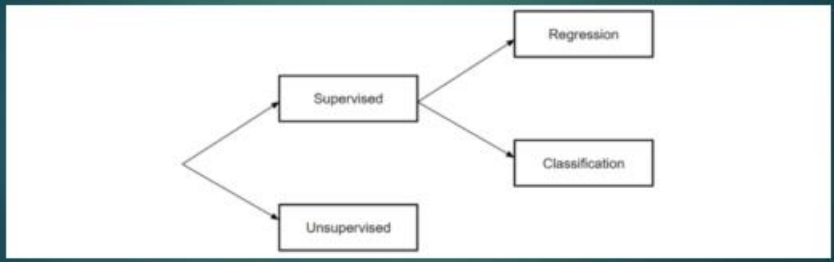
Шар згортки – це основний блок згорткової нейронної мережі. Він включає в себе для кожного каналу свій фільтр, ядро згортки якого обробляє попередній шар за фрагментами (підсумовуючи результати матричного добутку для кожного фрагмента).



МОДЕЛІ МАШИННОГО НАВЧАННЯ

7

Всі моделі машинного навчання поділяються (рисунок 3.1) на навчання з учителем (supervised) і без вчителя (unsupervised). В першу категорію входять регрессионная і класифікаційна моделі.

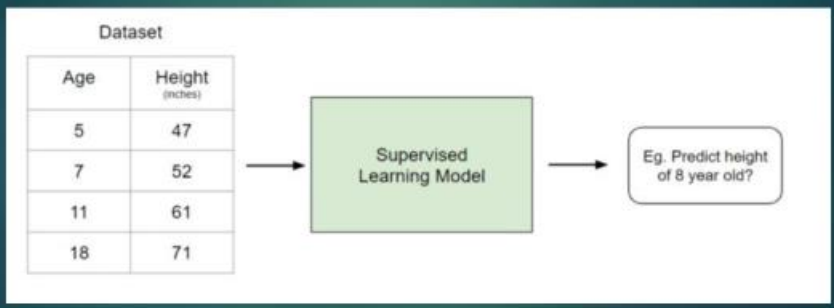


НАВЧАННЯ З ВЧИТЕЛЕМ

8

Навчання з вчиталем являє собою вивчення функції, яка перетворює вхідні дані у вихідні на основі прикладів пар введення-виведення.

Навчання з учителем підрозділяється на дві підкатегорії: регресія і класифікація.

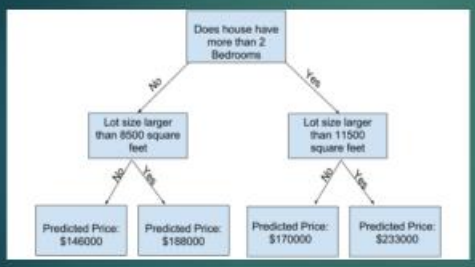


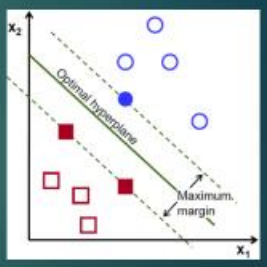
НАВЧАННЯ З ВЧИТЕЛЕМ

9

У регресійних моделях висновок є безперервним. Інують наступні типи регресійних моделей: лінійна регресія, дерево рішень, випадковий ліс, нейронна мережа.

У класифікаційних моделей висновок є дискретним. Найбільш поширені наструпні типи моделей: логістичная регресія, метод опорних векторів та інші.

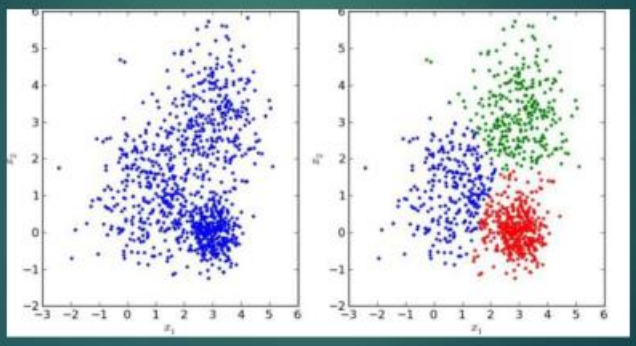




НАВЧАННЯ БЕЗ ВЧИТЕЛЯ

10

На відміну від навчання з учителем, навчання без вчителя використовується для того, щоб зробити висновки і знайти шаблони з вхідних даних без відсилань на помічені результати. Два основні методи, які використовуються в навчанні без учителя, включають кластеризацію і зниження розмірності.

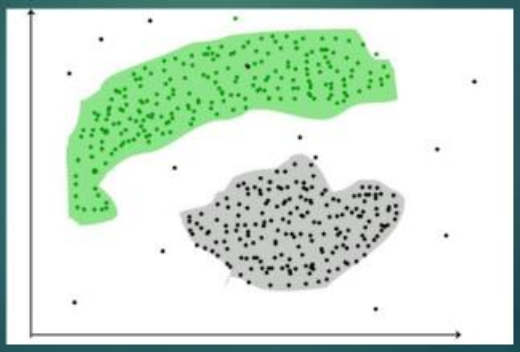


НАВЧАННЯ З ВЧИТЕЛЕМ

11

Кластеризація - це техніка навчання без учителя, яка включає в себе групування або кластеризацію точок даних. Найчастіше вона використовується для сегментації споживачів, виявлення шахрайства та класифікації документів.

Зниження розмірності - це процес зменшення числа розглянутих випадкових змінних шляхом отримання набору головних змінних. Простіше кажучи, це процес зменшення розміру набору ознак (зменшення кількості ознак).



ТЕХНОЛОГІЇ

12

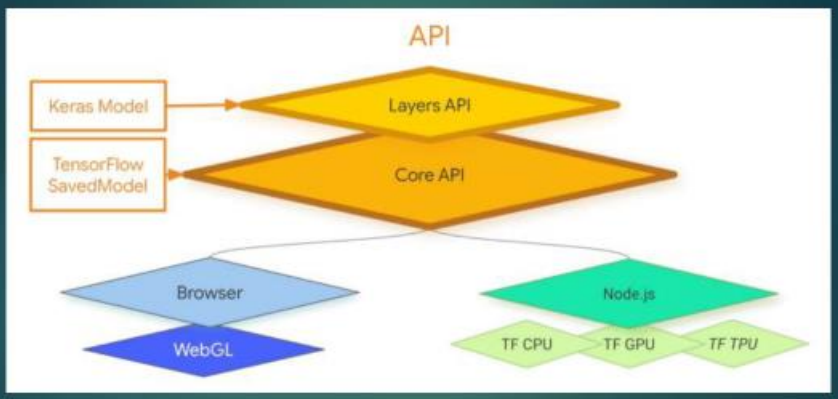
Для того щоб реалізувати весь програму для генерації правдоподібних зображень для навчання класифікаційних моделй було використано наступну технологію TensorFlow.js.

TensorFlow.js — відкрита програмна бібліотека для машинного навчання цілій низці задач, розроблена компанією Google для задоволення її потреб у системах, здатних будувати та тренувати нейронні мережі для виявлення та розшифровування образів та кореляцій, аналогічно до навчання й розуміння, які застосовують люди.

API TensorFlow.js можна використовувати для побудови моделей за допомогою бібліотеки лінійної алгебри JavaScript JavaScript або шарів API більш високого рівня. Перетворювачі моделі TensorFlow.js можуть запускати існуючі моделі в браузері або під Node.js. Існуючі моделі можна перекваліфікувати за допомогою даних датчиків, підключених до браузера.

API TensorFlow.js

13



АРХІТЕКТУРА МОДЕЛІ

14

Model Architecture

Layer Name	Output Shape	# Of Params	Trainable
conv2d_Conv2D1	[batch,24,24,8]	208	true
max_pooling2d_MaxPooling2D1	[batch,12,12,8]	0	true
conv2d_Conv2D2	[batch,8,8,16]	3,216	true
max_pooling2d_MaxPooling2D2	[batch,4,4,16]	0	true
flatten_Flatten1	[batch,256]	0	true
dense_Dense1	[batch,10]	2,570	true

РЕЗУЛЬТАТИ

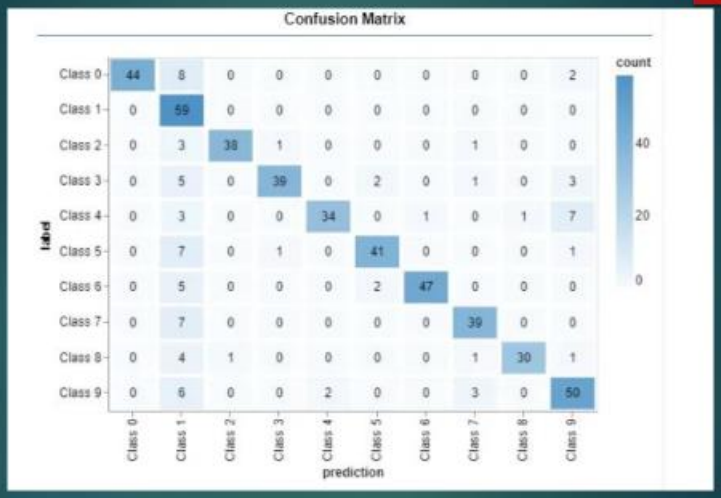
15

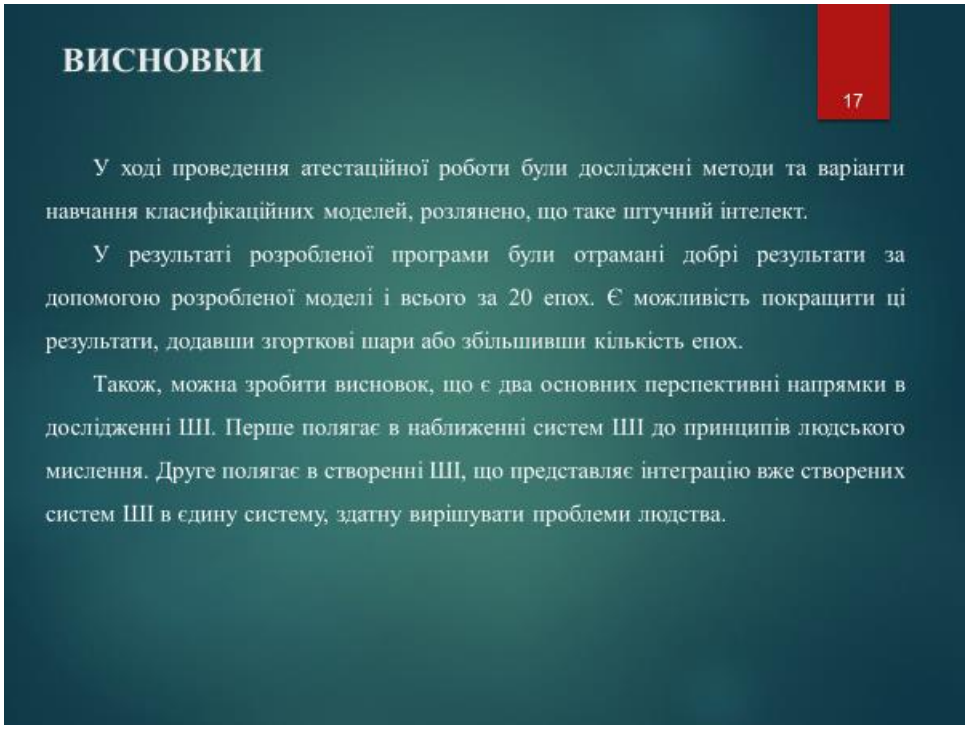
Accuracy

Class	Accuracy	# Samples
Zero	0.8158	38
One	0.9831	59
Two	0.8421	57
Three	0.7895	38
Four	0.8246	57
Five	0.8043	46
Six	0.7826	46
Seven	0.8667	60
Eight	0.7708	48
Nine	0.7255	51

РЕЗУЛЬТАТИ

16




ВИСНОВКИ
17
У ході проведення атестаційної роботи були досліджені методи та варіанти навчання класифікаційних моделей, розлянено, що таке штучний інтелект.
У результаті розробленої програми були отрамані добрі результати за допомогою розробленої моделі і всього за 20 епох. Є можливість покращити ці результати, додавши згорткові шари або збільшивши кількість епох.
Також, можна зробити висновок, що є два основних перспективні напрямки в дослідженні ШІ. Перше полягає в наближенні систем ШІ до принципів людського мислення. Друге полягає в створенні ШІ, що представляє інтеграцію вже створених систем ШІ в єдину систему, здатну вирішувати проблеми людства.

ДОДАТОК Б

Код програми

Б.1 Файл index.html

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>TensorFlow.js Convolutional Neural Networks</title>

  <!-- Import TensorFlow.js -->
  <script
src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@1.0.0/dist/tf.min.js"></sc
ript>
  <!-- Import tfjs-vis -->
  <script            src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs-
vis@1.0.2/dist/tfjs-vis.umd.min.js"></script>
          <!-- Import the data file -->
  <script src="./data.js" type="module"></script>

  <!-- Import the main script file -->
  <script src="./script.js" type="module"></script>
</head>

<body>
</body>
</html>
```

Б.2 Файл data.js

```
const IMAGE_SIZE = 784;
const NUM_CLASSES = 10;
const NUM_DATASET_ELEMENTS = 65000;

const TRAIN_TEST_RATIO = 5 / 6;

const     NUM_TRAIN_ELEMENTS     =     Math.floor(TRAIN_TEST_RATIO     *
NUM_DATASET_ELEMENTS);
const NUM_TEST_ELEMENTS = NUM_DATASET_ELEMENTS - NUM_TRAIN_ELEMENTS;

const MNIST_IMAGES_SPRITE_PATH =
    'https://storage.googleapis.com/learnjs-data/model-
builder/mnist_images.png';
const MNIST_LABELS_PATH =
    'https://storage.googleapis.com/learnjs-data/model-
builder/mnist_labels_uint8';

/**
 * A class that fetches the sprited MNIST dataset and returns shuffled
batches.
```

```
       *
       * NOTE: This will get much easier. For now, we do data fetching and
       * manipulation manually.
       */
      export class MnistData {
        constructor() {
          this.shuffledTrainIndex = 0;
          this.shuffledTestIndex = 0;
        }

        async load() {
          // Make a request for the MNIST sprited image.
          const img = new Image();
          const canvas = document.createElement('canvas');
          const ctx = canvas.getContext('2d');
          const imgRequest = new Promise((resolve, reject) => {
            img.crossOrigin = '';
            img.onload = () => {
              img.width = img.naturalWidth;
              img.height = img.naturalHeight;

              const datasetBytesBuffer =
                  new ArrayBuffer(NUM_DATASET_ELEMENTS * IMAGE_SIZE * 4);

              const chunkSize = 5000;
              canvas.width = img.width;
              canvas.height = chunkSize;

              for (let i = 0; i < NUM_DATASET_ELEMENTS / chunkSize; i++) {
                const datasetBytesView = new Float32Array(
                    datasetBytesBuffer, i * IMAGE_SIZE * chunkSize * 4,
                    IMAGE_SIZE * chunkSize);
                ctx.drawImage(
                    img, 0, i * chunkSize, img.width, chunkSize, 0, 0,
img.width,
                    chunkSize);

                const imageData = ctx.getImageData(0, 0, canvas.width,
canvas.height);

                for (let j = 0; j < imageData.data.length / 4; j++) {
                  // All channels hold an equal value since the image is
grayscale, so
                  // just read the red channel.
                  datasetBytesView[j] = imageData.data[j * 4] / 255;
                }
              }
              this.datasetImages = new Float32Array(datasetBytesBuffer);

              resolve();
            };
            img.src = MNIST_IMAGES_SPRITE_PATH;
          });

          const labelsRequest = fetch(MNIST_LABELS_PATH);
          const [imgResponse, labelsResponse] =
              await Promise.all([imgRequest, labelsRequest]);

          this.datasetLabels = new Uint8Array(await
labelsResponse.arrayBuffer());

          // Create shuffled indices into the train/test set for when we
select a
          // random dataset element for training / validation.
```

```
          this.trainIndices                                                    =
tf.util.createShuffledIndices(NUM_TRAIN_ELEMENTS);
          this.testIndices = tf.util.createShuffledIndices(NUM_TEST_ELEMENTS);

          // Slice the the images and labels into train and test sets.
          this.trainImages =
              this.datasetImages.slice(0, IMAGE_SIZE * NUM_TRAIN_ELEMENTS);
          this.testImages     =     this.datasetImages.slice(IMAGE_SIZE     *
NUM_TRAIN_ELEMENTS);
          this.trainLabels =
              this.datasetLabels.slice(0, NUM_CLASSES * NUM_TRAIN_ELEMENTS);
          this.testLabels =
              this.datasetLabels.slice(NUM_CLASSES * NUM_TRAIN_ELEMENTS);
        }

        nextDataBatch(batchSize, test = false) {
          if(test)
              return this.nextBatch(
                  batchSize, [this.trainImages, this.trainLabels], () => {
                    this.shuffledTrainIndex =
                        (this.shuffledTrainIndex        +        1)        %
this.trainIndices.length;
                    return this.trainIndices[this.shuffledTrainIndex];
                  });
          else
              return        this.nextBatch(batchSize,        [this.testImages,
this.testLabels], () => {
                  this.shuffledTestIndex =
                      (this.shuffledTestIndex + 1) % this.testIndices.length;
                  return this.testIndices[this.shuffledTestIndex];
                });
        }

        nextBatch(batchSize, data, index) {
          const batchImagesArray = new Float32Array(batchSize * IMAGE_SIZE);
          const batchLabelsArray = new Uint8Array(batchSize * NUM_CLASSES);

          for (let i = 0; i < batchSize; i++) {
            const idx = index();

            const image =
                data[0].slice(idx   *   IMAGE_SIZE,   idx   *   IMAGE_SIZE   +
IMAGE_SIZE);
            batchImagesArray.set(image, i * IMAGE_SIZE);

            const label =
                data[1].slice(idx   *   NUM_CLASSES,   idx   *   NUM_CLASSES   +
NUM_CLASSES);
            batchLabelsArray.set(label, i * NUM_CLASSES);
          }

          const xs = tf.tensor2d(batchImagesArray, [batchSize, IMAGE_SIZE]);
          const    labels    =    tf.tensor2d(batchLabelsArray,    [batchSize,
NUM_CLASSES]);

          return {xs, labels};
        }
      }
```

Б.3 Файл script.js

```
import {MnistData} from './data.js';

const getData = getDataFunction;
const createModel =  createModelFunction;
const trainModel = trainModelFunction;
const displayData = displayDataFunction;
const evaluateModel = evaluateModelFunction;

const classNames = ['Zero', 'One', 'Two', 'Three', 'Four', 'Five', 'Six', 'Seven', 'Eight', 'Nine'];

async function run() {
    const data = await getData();

    await displayDataFunction(data, 30);

    const model = createModel();
    tfvis.show.modelSummary({name: 'Model Architecture'}, model);

    await trainModel(model, data, 20);

    await evaluateModel(model, data);
}

/**
  * @desc retrieves data from defined location
  * @return wine data as json
*/
async function getDataFunction() {
    var data = new MnistData();
    await data.load();
    return data;
}

async function singleImagePlot(image)
{
  const canvas = document.createElement('canvas');
  canvas.width = 28;
  canvas.height = 28;
  canvas.style = 'margin: 4px;';
  await tf.browser.toPixels(image, canvas);
  return canvas;
}

async function displayDataFunction(data, numOfImages = 10) {

  const inputDataSurface =
    tfvis.visor().surface({ name: 'Input Data Examples', tab: 'Input Data'});

  const examples = data.nextDataBatch(numOfImages, true);

  for (let i = 0; i < numOfImages; i++) {
    const image = tf.tidy(() => {
      return examples.xs
        .slice([i, 0], [1, examples.xs.shape[1]])
        .reshape([28, 28, 1]);
    });

    const canvas = await singleImagePlot(image)
```

```
    inputDataSurface.drawArea.appendChild(canvas);

    image.dispose();
  }
}

function createModelFunction() {
  const cnn = tf.sequential();

  cnn.add(tf.layers.conv2d({
      inputShape: [28, 28, 1],
      kernelSize: 5,
      filters: 8,
      strides: 1,
      activation: 'relu',
      kernelInitializer: 'varianceScaling'
    }));
  cnn.add(tf.layers.maxPooling2d({poolSize: [2, 2], strides: [2, 2]}));

  cnn.add(tf.layers.conv2d({
      kernelSize: 5,
      filters: 16,
      strides: 1,
      activation: 'relu',
      kernelInitializer: 'varianceScaling'
    }));
  cnn.add(tf.layers.maxPooling2d({poolSize: [2, 2], strides: [2, 2]}));

  cnn.add(tf.layers.flatten());

  cnn.add(tf.layers.dense({
    units: 10,
    kernelInitializer: 'varianceScaling',
    activation: 'softmax'
  }));

  cnn.compile({
    optimizer: tf.train.adam(),
    loss: 'categoricalCrossentropy',
    metrics: ['accuracy'],
  });

  return cnn;
}

function getBatch(data, size, test = false)
{
    return tf.tidy(() => {
        const d = data.nextDataBatch(size, test);
        return [
          d.xs.reshape([size, 28, 28, 1]),
          d.labels
        ];
      });
}

async function trainModelFunction(model, data, epochs) {
    const metrics = ['loss', 'val_loss', 'acc', 'val_acc'];
    const container = {
      name: 'Model Training', styles: { height: '1000px' }
    };
    const fitCallbacks = tfvis.show.fitCallbacks(container, metrics);

    const batchSize = 512;
```

```
        const [trainX, trainY] = getBatch(data, 5500);
        const [testX, testY] = getBatch(data, 1000, true);

        return model.fit(trainX, trainY, {
          batchSize: batchSize,
          validationData: [testX, testY],
          epochs: epochs,
          shuffle: true,
          callbacks: fitCallbacks
        });
      }


      function predict(model, data, testDataSize = 500) {
        const testData = data.nextDataBatch(testDataSize, true);
        const testxs = testData.xs.reshape([testDataSize, 28, 28, 1]);
        const labels = testData.labels.argMax([-1]);
        const preds = model.predict(testxs).argMax([-1]);

        testxs.dispose();
        return [preds, labels];
      }


      async function displayAccuracyPerClass(model, data) {
        const [preds, labels] = predict(model, data);
        const  classAccuracy  =  await  tfvis.metrics.perClassAccuracy(labels,
preds);
        const container = {name: 'Accuracy', tab: 'Evaluation'};
        tfvis.show.perClassAccuracy(container, classAccuracy, classNames);

        labels.dispose();
      }

      async function displayConfusionMatrix(model, data) {
        const [preds, labels] = predict(model, data);
        const  confusionMatrix  =  await  tfvis.metrics.confusionMatrix(labels,
preds);
        const container = {name: 'Confusion Matrix', tab: 'Evaluation'};
        tfvis.render.confusionMatrix(
            container, {values: confusionMatrix}, classNames);

        labels.dispose();
      }

      async function evaluateModelFunction(model, data)
      {
        await displayAccuracyPerClass(model, data);
        await displayConfusionMatrix(model, data);
      }

      document.addEventListener('DOMContentLoaded',                    run);
```

№	Позначення	Найменування	Дод. відомості
		Текстові документи	
1		Пояснювальна записка	89 с.
		Графічні матеріали	
2		Слайд-презентація	17 слайдів
3		Компакт-диск (CD)	1 шт.

Змін.	Арк.	Номер докум.	Підп.	Дата	ГЮІК.ХХХХХХ.ХХХ ВД		
Розроб.		Колесников О.В.			Проектування комп'ютерної мережі малого підприємства. Засоби захисту інформації	Літ	Аркуш 1 / Аркушів 1
Перевір.		Руденко О.Г.				У	
Н.контр.		Руденко О.Г.			Відомість атестаційної роботи	ХНУРЕ Кафедра КІТС	
Затв.							