Αναφορά Εργασίας Προγμένων Θεμάτων Ανάλυσης Δεδομενων

Σε αυτή την αναφορά, γίνεται ανάλυση των βημάτων της εργασίας εξαμήνου.

Προαπαιτούμενα για τρέξιμο εφαρμογής

Η εργασία αναπτύχθηκε με την γλώσσα προγραμματισμού Python 3 (3.7.7) με την χρήση των παρακάτω πακέτων, των οποίων η εγκατάσταση έγινε με το εργαλείο pip.

- tensorflow : 2.3.0 – Για την δημιουργεία νευρωνικών δικτύων.
- opencv-contrib-python : 4.2.0.32 – Για την επεξεργασία εικόνων.
- matplotlib : 3.3.1 – Για την εμφάνιση εικόνων απο το F-MNIST.
- Flask : 1.1.2 – Για την ανάπτυξη του back-end του web app.

Φόρτωμα δεδομένων

Για να φορτώσουμε τα δεδομένα του Fashion MNIST dataset, έγινε λήψη των αρχείων απο https://github.com/zalandoresearch/fashion-mnist τα οποία αποθηκέυτικαν στον φάκελο app/data/fashion. Στη συνέχεια όταν εκτελούμε το app/perform_experiments.py τα αρχεία φορτόνονται στη μνήμη με την χρήση της μεθόδου load_mnist() απο το αρχείο app/src/classifier/mnist_reader.py . Το προαναφερόμενο αρχείο αντιγράφτικε απο το repository του fashion-mnist. Αυτό απλα βρίσκει τα συμπιεσμένα αρχεία δεδομένων, τα αποσυμπιέζει και τα χωρίζει σε ανεξάρτητες μεταβλητές (image data) και σε εξαρτημένες (ετικέτες).

Στην εκτελεση του perform_experiments() απλα καλειται μεθοδος load_data(), η οποία καλεί την load_mnist() και τελος κανονικοποιεί τα δεδομενα.

```
# Load dataset and split it to train set and test set
xtrain, ytrain, xtest, ytest = load_dataset()
```

```
def load_dataset(path='data/fashion', normalize=True):
    """
    Uses the load_mnist method from mnist_reader.py to split f-mnist data into train/test data and labels
    :param path: path to dataset
    :return: xtrain, ytrain, xtest, ytest
    """

    xtrain, ytrain = mnist_reader.load_mnist(path, kind='train')  # Extract f-mnist data using load_mnist from the mnist repository
    xtest, ytest = mnist_reader.load_mnist(path, kind='t10k')  # https://github.com/zalandoresearch/fashion-mnist
    if normalize:
        # Normalize
        xtrain, xtest = xtrain / 255.0, xtest / 255.0
    return xtrain, ytrain, xtest, ytest
```

Αξιολόγηση νευρωνικού δικτύου

Αφού φορτώσουμε τα δεδομένα στη μνήμη, εκτελούμε κάθε τύπου αξιολόγηση καλώντας την συνάρτηση evaluate_nets() του αρχείου app/src/classifier/mnist_classifier.py . Η συνάρτηση αυτή δημιουργεί n νευρωνικά δίκτυα για κάθε ένα ερώτημα της εργασίας (διαφορετικού αριθμού hidden layer, διαφορετικό αριθμό νευρώνων, κλπ) και τα αξιολογεί με βάση, παλι, τα απαιτούμενα κριτήρια της εργασίας. Τα αποτελέσματα της αξιολόγησης αποθηκεύονται στο αρχείο app/data/experiment-results.txt μετά απο κάθε εκτέλεση της evaluate_nets(). Παρακάτω υπάρχει ο κώδικας για την παραγωγή ενός νευρονικού δικτύου, καθώς και ένα κομμάτι απο τον κώδικα της evaluate_nets(), που είναι υπεύθυνος για την αξιολόγηση δικτύων με διαφορετικό αριθμό hidden layer. Επιπλέον φαίνεται παράδειγμα εξόδου απο το αρχείο experiment-results.txt . Αναλυτικά δεδομένα για ολα τα πειράματα που έχουν τρέξει βρίσκονται στο app/data/experiment-results.txt . Να σημειωθεί πως η εκτέλεση των δικτύων είναι μη-αιτιοκρατική, με αποτέλεσμα τα δεδομένα των πειραμάτων να μην είναι ίδια κάθε φορά.

```python
def generate_net(hidden_layers=2, n_neurons=128):
    """
    This method is used to generate a Sequential model using given parameters
    :param hidden_layers: Number of hidden layers
    :param n_neurons: Number of neurons per layer
    :return: Sequential model
    """
    # Initialize Sequential model
    model = tf.keras.models.Sequential()

    # Add input layer
    model.add(tf.keras.layers.Flatten(input_shape=(28, 28)))

    # Add hidden layers
    for i in range(hidden_layers):
        model.add(tf.keras.layers.Dense(n_neurons, activation='relu'))
    # Add output layer
    model.add(tf.keras.layers.Dense(10))

    model.summary()

    return model
```

```python
file = open('data/experiment-results.txt', 'a')
# Set training parameters
h_layers = 2
n_neurons = 128
epochs = 10

# Examine different hidden layer configurations
if eval_layers:
    file.write('-------------------------------- Layer Evaluation - | 128 neurons per layer | 10 epochs | 60000 training samples ----------------------\n')
    for h_layers in range(n_nets):
        model = generate_net(hidden_layers=h_layers + 1)  # Initialize network
        acc, train_time, average_inf_time = train_and_infer(model, x_train, y_train, x_test, y_test, epochs)  # Train and infer network
        file.write(
            'For {} hidden layers, the network achieved {:.2f}% classification accuracy, with {:.2f}s training time and {:.2f}s average inference time\n'.format(h_layers + 1,
                                                                                                                                                            acc * 100,
                                                                                                                                                            train_time,
                                                                                                                                                            average_inf_time))
    file.write('-------------------------------------------------------------------------------------------------------------------------------------\n')
    model = None
    h_layers = 2
```

Κώδικας 1: Τμήμα κώδικα της evaluate_nets()

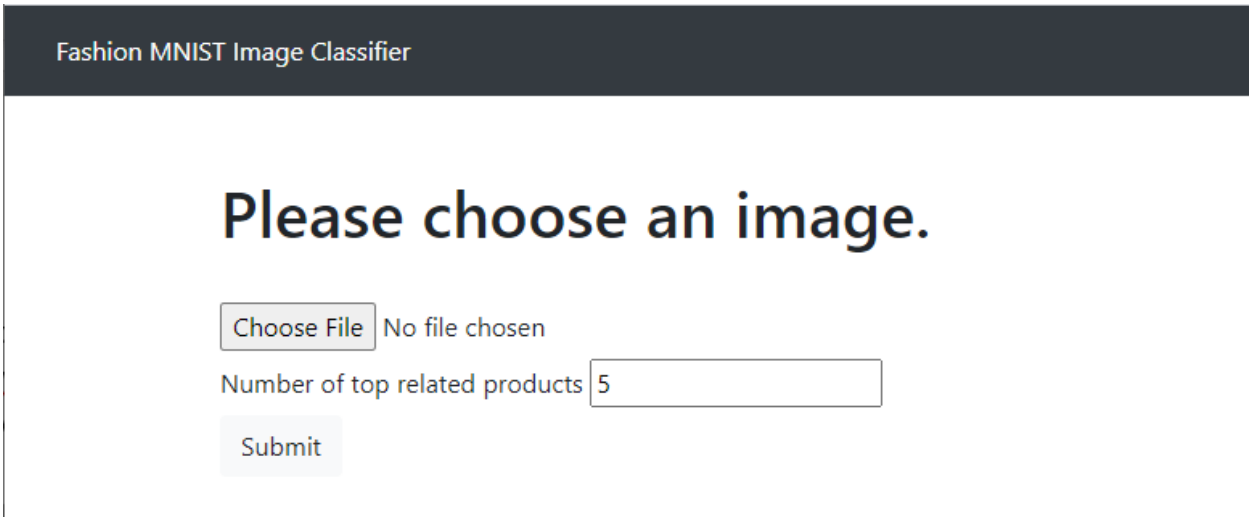
```
-------------------------------- Layer Evaluation - | 128 neurons per layer | 10 epochs | 60000 training samples ----------------------
For 1 hidden layers, the network achieved 87.33% classification accuracy, with 18.61s training time and 0.02s average inference time
For 2 hidden layers, the network achieved 88.44% classification accuracy, with 20.67s training time and 0.03s average inference time
For 3 hidden layers, the network achieved 87.75% classification accuracy, with 25.19s training time and 0.02s average inference time
For 4 hidden layers, the network achieved 87.88% classification accuracy, with 24.91s training time and 0.03s average inference time
For 5 hidden layers, the network achieved 88.18% classification accuracy, with 29.23s training time and 0.02s average inference time
For 6 hidden layers, the network achieved 88.19% classification accuracy, with 32.86s training time and 0.03s average inference time
For 7 hidden layers, the network achieved 88.17% classification accuracy, with 30.30s training time and 0.02s average inference time
For 8 hidden layers, the network achieved 87.82% classification accuracy, with 36.71s training time and 0.03s average inference time
For 9 hidden layers, the network achieved 87.07% classification accuracy, with 37.31s training time and 0.02s average inference time
For 10 hidden layers, the network achieved 87.79% classification accuracy, with 37.64s training time and 0.02s average inference time
-------------------------------------------------------------------------------------------------------------------------------------
```

Ανάπτυξη εφαρμογής

Η εφαρμογή είναι web-based και μπορεί να τρέξει με την εκτέλεση του αρχείου app/init_webservice.py . Η εφαρμογή θα τρέξει στο http://localhost:5000/. Για την ανάπτυξη της εφαρμογής έγινε η χρήση του Flask module της python. Η εφαρμογή θα είναι έτοιμη για χρήση μόλις εμφανιστεί το παρακάτω μήνυμα στο τερματικό:

```
* Debugger is active!
* Debugger PIN: 301-071-546
* Running on http://localhost:5000/ (Press CTRL+C to quit)
```

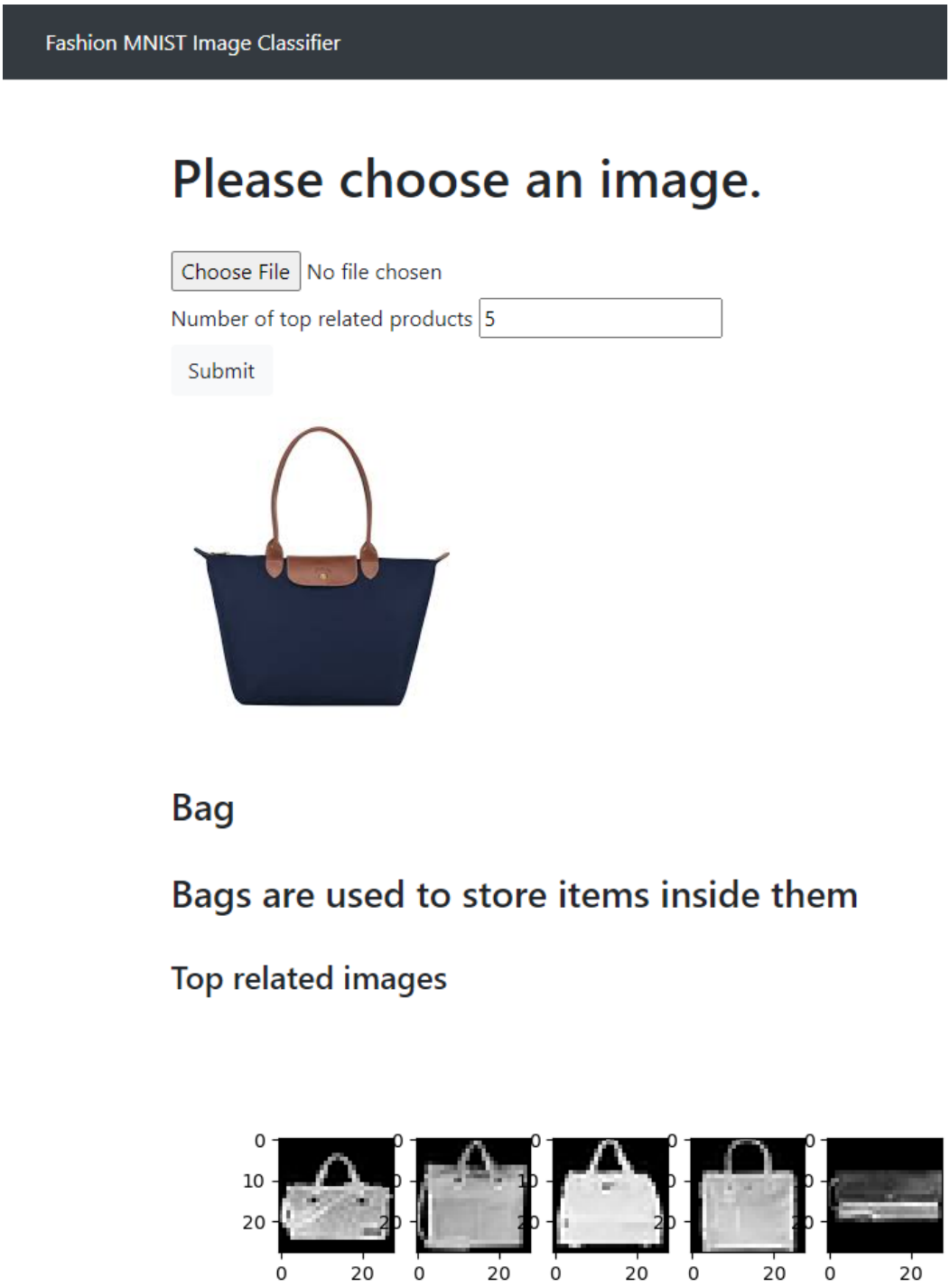
Μόλις ανοίξουμε την εφαρμογή, θα πρέπει να εμφανίζεται η παρακάτω σελίδα:



Επιλέγουμε μια εικόνα πατόντας το κουμπί ‘Choose File’, και συνεχίζουμε επιλέγοντας παραμέτρους. Αφου πατήσουμε ‘Submit’ εκτελείται η συνάρτηση process_image() του αρχείου app/src/classification/classification_handler.py , για να έρθει η φωτογραφία εισόδου στις διαστάσεις των φωτογραφιών που αποτελούν το Fashion-MNIST, και μετα εκτελούμε το προ-εκπαιδευμένο μοντέλο για να κατηγοριοποιήσουμε την εικόνα. Να ληφθεί υπ’ όψην πως για την εφαρμογή χρησιμοιήθηκε συνελικτικό

νευρονικό δίκτυο, καθώς αυτό, παρότι δεν είναι τέλειο, φάνικε πιό κατάληλο για την κατηγοριοποίηση εικόνων.

Αφού γίνει επιτυχώς η κατηγοριοποίηση της εικόνας, θα πρεπει να εχουμε ενα παρόμοιο αποτέλεσμα με το παρακάτω:



Οι ‘Top relate images’ είναι οι n φωτογραφίες με την μεγαλύτερη (τυχαία κατα την έναρξη της εφαρμογής) βαθμολογία. Υπεύθυνες συναρτήσεις για την ανάθεση και επιλογή αυτών των φωτογραφιών είναι οι : rank_images(), top_k_images() και save_top_images().

Ανακεφαλαίωση για τρέξιμο εργασίας

Αφού έχουν εγκατασταθεί το προαπαιτούμενα

- Για τρέξιμο και αξιολόγιση δικτύου:
  - cd FMISTNN/app/
  - python perform_experiments.py
  - Τα αποτελέσματα ειναι στο FMNIST/app/data/experiment-results.txt
- Για έναρξη εφαρμογής:
  - cd FMNIST/app/
  - python init_webservice.py