Integrated M.Sc. Data Science

18CSC402- Deep learning

Evaluation Lab 2

22-10-2021

CO-PO Mapping

CO1	Outline the basics of artificial neural networks
CO2	Understand the difficulty of training deep neural networks
CO3	Distinguish and select appropriate Neural Network training mechanisms.
CO4	Compare and contrast Recurrent and Convolutional Neural networks
CO5	Apply the deep learning concepts in Natural Language Processing and Computer Vision domains

---

TAMIL CHARACTER CLASSIFICATION

a) Individual team members contribution:

Roll. No	Contribution
CB.SC.I5DAS18009	Data pre-processing , CNN Architecture - 1
CB.SC.I5DAS18022	Data Augmentation ,CNN Architecture - 2
CB.SC.I5DAS18026	Data Augmentation, CNN Architecture - 3
CB.SC.I5DAS18033	CNN Architecture - 4
CB.SC.I5DAS18042	Data pre-processing, CNN Architecture - 5

b) Few words about Dataset:

- ➔ Language: Tamil
- ➔ Dataset url: https://drive.google.com/file/d/1fnfdaAAQA-v1l7USrpJAKSWzx3fD0ln8/view?usp=sharing
- ➔ Characters chosen: ன ஏ ஐ ஒ ஓ ஒள
- ➔ Size of the data set: 1456

Data Pre-processing:

- ✔ After loading the data, we converted all the images from .tiff to .jpeg format.
- ✔ Since the distribution of characters was uneven, we performed ***data augmentation*** for the last character.
- ✔ We generated data ourselves for the last character since the original data had only one sample.
- ✔ We resized all the images to the same size - (224,224).
- ✔ We classified the samples and stored them in different folders according to their characters.

Encoding:

- We used label and one-hot encoding to identify the 6 classes separately. Then used one hot encoder for the rest of the data.

Train-Test split:

- ➔ Size of train data: 975
- ➔ Size of test data: 481
- ➔ Size of validation data: 481

CNN Model:

Each of us developed different CNN architecture with the image data set and predicted the classes.

c) Results and observation:

CB.SC.I5DAS18009 -

```
model1 = Sequential()
model1.add(Conv2D(32, (5, 5), input_shape=(128,128,1), activation='relu'))
model1.add(MaxPooling2D(pool_size=(2, 2)))
model1.add(Conv2D(16, (5, 5), activation='relu'))
model1.add(MaxPooling2D(pool_size=(2, 2)))
model1.add(Conv2D(16, (5, 5), activation='relu'))
model1.add(MaxPooling2D(pool_size=(2, 2)))
model1.add(Dropout(0.2))
model1.add(Flatten())
model1.add(Dense(64, activation='relu'))
model1.add(Dense(6, activation='softmax'))
model1.summary()
model1.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

	Epochs	Accuracy	Loss
Architecture 1	5	0.9569	0.1461
	10	0.9918	0.0316
	15	0.9969	0.0099
	Final	0.9780	0.1217

CB.SC.I5DAS18022 -

```
model2 = Sequential()
model2.add(Conv2D(128, (3, 3), input_shape=(128,128,1),padding='valid',strides=(1,1), activation='relu'))
model2.add(MaxPooling2D(pool_size=(2, 2)))
model2.add(Dropout(0.2))
model2.add(Conv2D(64, (3, 3), activation='relu'))
model2.add(MaxPooling2D(pool_size=(2, 2)))
model2.add(Conv2D(32, (3, 3), activation='relu'))
model2.add(MaxPooling2D(pool_size=(2, 2)))
model2.add(Dropout(0.2))
model2.add(Flatten())
model2.add(Dense(256, activation='relu'))
model2.add(Dense(6, activation='softmax'))
model2.summary()
model2.compile(loss='categorical_crossentropy', optimizer='Adam', metrics=['accuracy'])
```

	Epochs	Accuracy	Loss
Architecture 2	5	0.9692	0.0853
	10	0.9789	0.0931
	13	0.9969	0.0069
	Final	0.9760	0.1269

CB.SC.I5DAS18026 -

```
model3 = Sequential()
model3.add(Conv2D(32, (5, 5), activation='relu', strides=(2, 2), padding='same', input_shape=(128, 128, 1)))
model3.add(MaxPooling2D(pool_size=(2, 2)))
model3.add(Conv2D(50, (5, 5), activation='relu', padding='same'))
model3.add(MaxPooling2D(pool_size=(2, 2)))
model3.add(Conv2D(70, (3, 3), activation='relu', padding='same'))
model3.add(MaxPooling2D(pool_size=(2, 2)))
model3.add(Flatten())
model3.add(Dense(100, activation='relu'))
model3.add(Dropout(0.1))
model3.add(Dense(6, activation='softmax'))
model3.summary()
model3.compile(loss = 'binary_crossentropy', optimizer = 'rmsprop', metrics = ['accuracy'])
```

	Epochs	Accuracy	Loss
Architecture 3	5	0.8903	0.1149
	10	0.9713	0.0354
	15	0.9815	0.0180
	Final	0.9602	0.0398

CB.SC.I5DAS18033 -

```
model4 = Sequential()
model4.add(Conv2D(128, (5, 5), activation='relu',  padding='valid', input_shape=(128, 128, 1)))
model4.add(MaxPooling2D(pool_size=(2, 2)))
model4.add(Conv2D(64, (5, 5), activation='relu', padding='valid'))
model4.add(MaxPooling2D(pool_size=(2, 2)))
model4.add(Conv2D(32, (3, 3), activation='relu', padding='valid'))
model4.add(MaxPooling2D(pool_size=(2, 2)))
model4.add(Conv2D(16, (3, 3), activation='relu', padding='valid'))
model4.add(MaxPooling2D(pool_size=(2, 2)))
model4.add(Flatten())
model4.add(Dense(100, activation='relu'))
model4.add(Dropout(0.4))
model4.add(Dense(6, activation='sigmoid'))
model4.summary()
model4.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
```

	Epochs	Accuracy	Loss
Architecture 4	5	0.7836	0.6394
	8	0.8677	0.3648
	12	0.9282	0.2106
	Final	0.9396	0.1672

CB.SC.I5DAS18042 -

```
model5 = Sequential()
model5.add(Conv2D(64, (5, 5), input_shape=(128,128,1), activation='relu'))
model5.add(MaxPooling2D(pool_size=(2, 2)))
model5.add(Dropout(0.2))
model5.add(Conv2D(32, (5, 5), activation='relu'))
model5.add(MaxPooling2D(pool_size=(2, 2)))
model5.add(Conv2D(32, (5, 5), activation='relu'))
model5.add(MaxPooling2D(pool_size=(2, 2)))
model5.add(Dropout(0.2))
model5.add(Conv2D(32, (5, 5), activation='relu'))
model5.add(MaxPooling2D(pool_size=(2, 2)))
model5.add(Dropout(0.2))
model5.add(Flatten())
model5.add(Dense(256, activation='relu'))
model5.add(Dense(6, activation='softmax'))
model5.summary()
model5.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

	Epochs	Accuracy	Loss
Architecture 5	5	0.9067	0.2825
	10	0.9600	0.0983
	15	0.9723	0.0672
	Final	0.9808	0.0669

d) Overall Observation:

The data responded well with all the architectures and showed improvement while increasing epochs gradually. Best CNN model was Architecture 5 which gave the accuracy of 98% !!