**Deep Learning and Neural Networking-based Life Science research**

An original report submitted in partial fulfillment of the requirements for the 3-month advanced in silico research training program in the
**Deep Learning and Neural Networking-based Life Science research**

By
Dharun Ramesh

@
**Quick IsCool**
Aitele Research LLP

## Certificate

This is to certify that the work contained in this report entitled

**Stage 1: Introduction to Biological Data Analysis**

submitted by

**Dharun Ramesh**

towards the partial requirement of the successful completion of the Scientific Training Program at Quick IsCool has been carried out under the consultancy and guidance of **Quick IsCool Experts** and **Artificial Intelligence-based Assistant Bots** and that it has not been submitted elsewhere for the award of any degree or diploma.

# Experiment No. 1: Introduction to Biological Data Analysis

## Introduction to the Basic Concepts

In this experiment, we explore the fundamentals of biological data analysis by acquiring and preprocessing basic biological datasets, specifically gene expression data. The goal is to implement a simple neural network for classifying samples based on biological features.

## Important URLs

https://paperswithcode.com/task/histopathological-image-classification
https://bmirds.github.io/MHIST/

## A Step-wise Protocol for the Experiment

1. Data Acquisition:

Download the gene expression dataset from the provided URL.

Load the dataset using appropriate libraries (e.g., pandas).

2. Data Preprocessing:

Explore and understand the structure of the dataset.

Clean and preprocess the data, handling any missing values or outliers.

3. Neural Network Implementation:

Use TensorFlow and Keras to build a simple neural network.

Define the architecture, compile the model, and set up training parameters.

4. Model Training:

Split the dataset into training and validation sets.

Train the neural network on the training set.

5. Evaluation and Visualization:

Evaluate the model performance on the validation set.

Visualize training and validation loss, accuracy, and other relevant metrics.

6. Conclusion:

Summarize the findings and insights gained from the experiment.

## **Programming & Codes** (If any)

```
Found 195 images belonging to 2 classes.
```

In [7]:
```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(2, activation='sigmoid')
])
```

In [8]:
```
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

In [9]:
```
history = model.fit(train_generator,
                    epochs=2,
                    validation_data=validation_generator)
```

```
Epoch 1/2
```

In [2]:
```
ano = pd.read_csv("C:\\Users\\dharu\\Downloads\\annotations.csv")
```

In [3]:
```
ano.head()
```

Out[3]:

| | Image Name | Majority Vote Label | Number of Annotators who Selected SSA (Out of 7) | Partition |
|---|---|---|---|---|
| 0 | MHIST_aaa.png | SSA | 6 | train |
| 1 | MHIST_aab.png | HP | 0 | train |
| 2 | MHIST_aac.png | SSA | 5 | train |
| 3 | MHIST_aae.png | HP | 1 | train |
| 4 | MHIST_aaf.png | SSA | 5 | train |

In [ ]:
```
"""
for i in range(len(ano)):
    old_path = f"C:\\Users\\dharu\\OneDrive\\Desktop\\Quick_IsCool\\images\\{ano['Image Name'][i]}"
    new_path = f"C:\\Users\\dharu\\OneDrive\\Desktop\\Quick_IsCool\\{ano['Partition'][i]}\\{ano['Majority Vote Label'][i]}\\{ano[

    if not os.path.exists(old_path):
        print(f"File not found at {old_path}")
    else:
        try:
            shutil.move(old_path, new_path)
            print(f"File moved from {old_path} to {new_path}")
        except shutil.Error as e:
            print(f"Error occurred while moving the file: {e}")
"""
```

## **Google Colab/ Jupyter Notebook** (If any)

https://colab.research.google.com/drive/1raJlSSZ7ixBwvd5nvtXmQk2dwg0TKAtl

## Certificate

This is to certify that the work contained in this report entitled

**Stage 7: Survival Analysis in Cancer Research**

submitted by

**Dharun Ramesh**
towards the partial requirement of the successful completion of the Scientific Training Program at Quick IsCool has been carried out under the consultancy and guidance of **Quick IsCool Experts** and **Artificial Intelligence-based Assistant Bots** and that it has not been submitted elsewhere for the award of any degree or diploma.

# Experiment 7: Survival Analysis in Cancer Research

## Introduction to the Basic Concepts

Survival analysis is a crucial aspect of cancer research, aiming to predict patient outcomes based on various factors. In this experiment, the focus is on utilizing neural networks for survival analysis, specifically based on gene expression data.

The primary dataset used in this experiment is sourced from [The Cancer Genome Atlas (TCGA)](https://www.cancer.gov/ccg/research/genome-sequencing/tcga), providing valuable genomic information for cancer research.

## Important URLs

https://www.cancer.gov/ccg/research/genome-sequencing/tcga

## A Step-wise Protocol for the Experiment

Data Loading and Preprocessing:

- The dataset is loaded from the file "preprocessed.csv."
- Irrelevant columns "exposures" and "diagnoses" are dropped.

Data Labeling and Encoding:

- Object-type columns are label-encoded for compatibility with machine learning algorithms.

Feature Selection:

- Correlation analysis is performed, and features are selected based on SelectKBest with ANOVA F-statistic.

Standardization:

- Features are standardized using StandardScaler.

Model Training and Evaluation:

- Random Forest and XGBoost classifiers are trained and evaluated.

## Programming & Codes

```
In [31]: f = SelectKBest(score_func = f_classif,k = len(features))
         feat_f = f.fit_transform(X,y)
         featf = X.columns[f.get_support()]
         f_X = X[featf]

Out[31]: Index(['case_id', 'race', 'gender', 'ethnicity', 'age_at_index',
                'submitter_id', 'days_to_birth', 'year_of_birth', 'demographic_id',
                'updated_datetime'],
               dtype='object')

In [33]: standard = StandardScaler()
         st_X = standard.fit_transform(f_X)

In [43]: ran = RandomForestClassifier()
         X_train,X_test,y_train,y_test = train_test_split(X,y)

In [44]: ran.fit(X_train,y_train)

Out[44]: RandomForestClassifier()

In [50]: print(classification_report(y_test,ran.predict(X_test)))
         print("Confusion Matrix\n",confusion_matrix(y_test,ran.predict(X_test)))

                      precision    recall  f1-score   support

                   0       0.76      0.81      0.79        16
                   1       0.00      0.00      0.00         4

            accuracy                           0.65        20
           macro avg       0.38      0.41      0.39        20
        weighted avg       0.61      0.65      0.63        20
```

## Google Colab/ Jupyter Notebook

https://colab.research.google.com/drive/18_SDFSOxqiIn-M6PZijS7YwhOe_q-DBo

## Certificate

This is to certify that the work contained in this report entitled

**Stage 5: Image Analysis in Pathology**

submitted by

**Dharun Ramesh**

towards the partial requirement of the successful completion of the Scientific Training Program at Quick IsCool has been carried out under the consultancy and guidance of **Quick IsCool Experts** and **Artificial Intelligence-based Assistant Bots** and that it has not been submitted elsewhere for the award of any degree or diploma.

# Experiment No. 5: Image Analysis in Pathology

## Introduction to the Basic Concepts

Histopathological image classification is a crucial task in cancer diagnosis. This experiment employs a Convolutional Neural Network (CNN) to classify histopathological images into five categories: Lung benign tissue, Lung adenocarcinoma, Lung squamous cell carcinoma, Colon adenocarcinoma, and Colon benign tissue. The dataset, LC25000, comprises 25,000 images of lung and colon tissues, each with five classes. The images were generated from HIPAA compliant and validated sources, with augmentation using the Augmentor package.

## Important URLs

- Original Article
- GitHub Repository

## A Step-wise Protocol for the Experiment

1. Import necessary libraries and modules.
2. Define an ImageDataGenerator for data augmentation.
3. Create separate generators for training and validation sets using flow_from_directory.
4. Build a CNN model with Conv2D, MaxPooling2D, Flatten, and Dense layers.
5. Compile the model with 'adam' optimizer and 'categorical_crossentropy' loss.
6. Train the model on the training generator with validation data.
7. Plot training and validation loss over epochs.
8. Plot training and validation accuracy over epochs.

## Programming & Codes (If any)

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
import matplotlib.pyplot as plt
```

In [2]:
```
datagen = ImageDataGenerator(
    rescale=1.0 / 255,
    validation_split=0.2
)
```

In [3]:
```
train_generator = datagen.flow_from_directory(
    "C:/Users/dharu/OneDrive/Desktop/Quick IsCool/Experiment5/lung_colon_image_set",
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='training'
)
```

```
Found 20000 images belonging to 2 classes.
```

In [4]:
```
valid_generator = datagen.flow_from_directory(
    "C:/Users/dharu/OneDrive/Desktop/Quick IsCool/Experiment5/lung_colon_image_set",
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='training'
)
```

```
Found 20000 images belonging to 2 classes.
```

```
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

In [*]:
```
history = model.fit(train_generator,
                    epochs=2,
                    validation_data=valid_generator)
```

```
Epoch 1/2
347/625 [===============>.............] - ETA: 14:02 - loss: 0.8123 - accuracy: 0.5930
```

In [ ]:
```
plt.plot(history.history["loss"],label = "Training Loss")
plt.plot(history.history["val_loss"],label = "Validation loss")
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend()
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.show()
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend()
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.show()
```

## Google Colab/ Jupyter Notebook

https://colab.research.google.com/drive/1HnKuBDYDidlh14wmmlprPu4ZdqmyvWYR

## Abbreviations

- CNN: Convolutional Neural Network
- LC25000: Lung and Colon Cancer Histopathological Image Dataset

## References


- Borkowski AA, Bui MM, Thomas LB, Wilson CP, DeLand LA, Mastorides SM. Lung and Colon Cancer Histopathological Image Dataset (LC25000). arXiv:1912.12142v1 [eess.IV], 2019

# Certificate

This is to certify that the work contained in this report entitled

**Stage 2: Basic Gene Expression Analysis**

submitted by

**Dharun Ramesh**

towards the partial requirement of the successful completion of the Scientific Training Program at Quick IsCool has been carried out under the consultancy and guidance of **Quick IsCool Experts** and **Artificial Intelligence-based Assistant Bots** and that it has not been submitted elsewhere for the award of any degree or diploma.

## Experiment No. 2: Basic Gene Expression Analysis

## Introduction to the Basic Concepts

This experiment involves a basic gene expression analysis using linear regression models. The dataset originates from a proof-of-concept study by Golub et al. in 1999, demonstrating the classification of cancer cases based on gene expression monitoring via DNA microarrays. The data was used to classify patients with acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL). The primary goal is to analyze gene expression patterns and visualize changes in response to different conditions.

## Important URLs

- Colab Notebook
- Original Study

## A Step-wise Protocol for the Experiment

1. Import necessary libraries including pandas, lifelines, and scikit-learn modules.
2. Load the gene expression dataset provided in CSV format.
3. Explore the dataset to understand its structure and content.
4. Prepare the data by selecting numerical columns for features and the 'call' column for the target variable.
5. Split the data into training and testing sets.
6. Encode categorical values in the 'call' column using one-hot encoding.
7. Train a linear regression model on the training set.
8. Make predictions on the test set and visualize gene expression changes using a scatter plot.

## Programming & Codes (If any)

```
from xgboost import XGBClassifier
from sklearn.compose import ColumnTransformer
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.feature_selection import SelectKBest,chi2,f_classif,SelectFpr,SelectFdr,S
import matplotlib.pyplot as plt
```

```
4]  data = pd.read_csv("/content/data_set_ALL_AML_train.csv")
```

```
5]  data.info()

    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 7129 entries, 0 to 7128
    Data columns (total 78 columns):
     #   Column                Non-Null Count  Dtype
    ---  ------                --------------  -----
     0   Gene Description      7129 non-null   object
```

```
[9]  X = data.iloc[:, 2::2]  # Selecting numerical columns
     y = data['call']        # Target variable (response to different conditions)
```

```
[10] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
[19] ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0])], remainder='passthrough')
     y_encoded = ct.fit_transform(pd.DataFrame(y))

     # Split the data into training and testing sets
     X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)
     model = LinearRegression()
     model.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
[22] predictions = model.predict(X_test)

     # Visualize gene expression changes
     plt.scatter(y_test, predictions)
     plt.xlabel('Actual Values')
```

## Google Colab/ Jupyter Notebook

https://colab.research.google.com/drive/1RwKSVuB1Md8b156iZAxBzY7WaF8noxxx

## Abbreviations

- AML: Acute Myeloid Leukemia
- ALL: Acute Lymphoblastic Leukemia

## References

Golub TR, et al. "Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring." Science 286:531-537. (1999).

## Certificate

This is to certify that the work contained in this report entitled

**Stage 3: Basic Protein Structural Analysis**

submitted by

**Dharun Ramesh**

towards the partial requirement of the successful completion of the Scientific Training Program at Quick IsCool has been carried out under the consultancy and guidance of **Quick IsCool Experts** and **Artificial Intelligence-based Assistant Bots** and that it has not been submitted elsewhere for the award of any degree or diploma.

# Experiment No. 3: Basic Protein Structural Analysis

## Introduction to the Basic Concepts

Protein structural analysis is a crucial aspect of understanding the functions of biological macromolecules. In this experiment, we delve into the fundamentals of protein structure prediction using data obtained from the Research Collaboratory for Structural Bioinformatics (RCSB) Protein Data Bank (PDB). The PDB archive serves as a repository for atomic coordinates and information about proteins, providing insights into their three-dimensional structures. Techniques such as X-ray crystallography and NMR spectroscopy contribute to determining the spatial arrangement of atoms in a molecule.

## Important URLs

- RCSB Protein Data Bank

## A Step-wise Protocol for the Experiment

1. Access the RCSB Protein Data Bank to retrieve protein structural data.
2. Download the provided dataset files: pdb_data_no_dups.csv and data_seq.csv.
3. Explore and preprocess the datasets, considering protein classification, extraction methods, and sequence information.
4. Utilize structural biophysical methods to understand the characteristics of protein structures.
5. Perform basic statistical and visual analyses to gain insights into the dataset.

## Programming & Codes (If any)

```
[ ] tokenizer = Tokenizer(char_level=True)
    tokenizer.fit_on_texts(sequences)
    sequences = tokenizer.texts_to_sequences(sequences)
    padded_sequences = pad_sequences(sequences)
```

```
[ ] X_train, X_test, y_train, y_test = train_test_split(padded_sequences, encoded_labels, test_size=0.2, random_state=42)
```

```
[ ] y_train_one_hot = to_categorical(y_train)
    y_test_one_hot = to_categorical(y_test)
```

```
[ ] model = Sequential()
    model.add(Embedding(input_dim=len(tokenizer.word_index) + 1, output_dim=50, input_length=padded_sequences.shape[1]))
    model.add(SimpleRNN(100))
    model.add(Dense(len(label_encoder.classes_), activation='softmax'))
```

```
[ ] model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

```
model.fit(X_train, y_train_one_hot, epochs=5, validation_data=(X_test, y_test_one_hot))

Epoch 1/5
 373/1627 [=====>.....................] - ETA: 34:10 - loss: 0.5867 - accuracy: 0.8585
```

## Google Colab/ Jupyter Notebook (If any)

https://colab.research.google.com/drive/1iqyMvQphWHPuGCzVXUeM_nIbxmOxZzVQ#scrollTo=uToJfTT6VZnK

## Abbreviations

- RCSB: Research Collaboratory for Structural Bioinformatics
- PDB: Protein Data Bank
- NMR: Nuclear Magnetic Resonance

## References

Original data set downloaded from RCSB Protein Data Bank