Michigan Tech

# SAT5114 R02 Final Project

## Chest Cancer Detection

**Team Members** – Shalaka Gaidhani, Tejaswi Chintapalli, Goutham Thota

**Participation** – We are contributing equally to all tasks.

**Problem Statement – The** clinical Problems of chest cancers, such as lung cancer and other malignancies, pose significant health risks globally, resulting in illness and death. Early detection plays a vital role in improving patient outcomes, as it enables timely interventions and potentially curative treatments. The development of accurate and efficient chest cancer detection models can enhance the chances of detecting cancers at an early stage when treatment options are most effective. This can potentially save lives and lead to improved patient prognosis, contributing to better health outcomes and quality of life for patients.

**Task** –

- Binary classification is the task to be performed which is nothing but employing a pre-trained neural network for feature extraction and a custom classifier for final classification to categorize chest CT scan images as malignant or non-cancerous.
- Identifying malignancies in chest CT scan pictures can help with the early identification and detection of lung cancer and other chest malignancies.

**Significance**:

- Algorithms for detecting chest cancer can be made more accurate and effective by utilizing transfer learning, which can benefit from knowledge from previously trained models.
- Radiologists and clinicians can make more informed judgments in clinical practice with the help of the development of precise and reliable models for the identification of chest cancer at an early stage.

**Dataset used -**

I. The data consists of around 1000 images of chest cancer types, namely Adenocarcinoma, Large cell carcinoma, and Squamous cell carcinoma, as well as a folder for normal cells.
II. The images are in jpg or png format, not in dcm format, in order to fit the model.
III. The data is organized into folders, with the main folder named "Data" which contains several sub-folders with sub-folders as "Data" folder for "test", "train", and "valid", which represents the testing, training, and validation sets respectively.
IV. The training set comprises 70% of the data, the testing set comprises 20% of the data, and the validation set comprises 10% of the data.

**Data Augmentation, splitting, and performance evaluation -**

I. We are using ImageDataGenerator class from the Keras library to perform data augmentation that generates batches of images during training.
II. The data is split into the train, validation, and test sets using three separate instances of ImageDataGenerator
III. The 'batch_size' parameter will be used which indicates that 32 images will be loaded and processed in each batch during training.
The 'target_size' parameter will be used to specify the size to which the images will be resized during preprocessing.
IV. Model evaluation metrics such as accuracy, precision, recall, F1-score will be used to predict the performance of each model.

**Proposed Models**

**1. CNN Model:**

I. Training data will be generated using ImageDataGenerator which will rescale pixel values to the range of 0 to 1.
II. The model architecture consists of Conv2D and MaxPooling2D layers with ReLU activation, Dropout layers for regularization, a Flatten layer, and Dense layers with ReLU and softmax activation for the output.
III. The model will be compiled with the Adam optimizer with a learning rate of 0.00001 and decay of 1e-5, categorical cross-entropy loss, and accuracy as the evaluation metric.
IV. The model will be trained using the fit method with a batch size of 32, for 10 epochs, and with callbacks for model check pointing and early stopping.

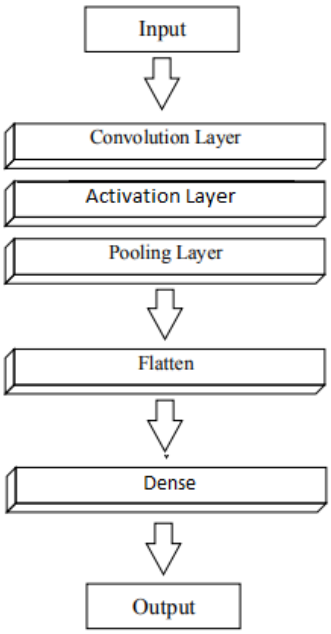


Fig:1 Proposed 2 Layer CNN Architecture

**2. ResNet50 Model:**

I. The training data will be generated using ImageDataGenerator with the rescaling of pixel values.
II. Input to the ResNet50 model will be pre-trained weights from ImageNet.
III. Conv5 layers will be made trainable.
IV. The model architecture consists of the ResNet50 model, followed by Dropout, Flatten, BatchNormalization, Dropout, and Dense layers with softmax activation for the output.
V. The model will be compiled with the Adam optimizer with a learning rate of 0.00001, a decay of 1e-6, categorical cross-entropy loss, and accuracy as the evaluation metric.
VI. The model will be trained using the fit method with a batch size of 32, for 10 epochs, and with callbacks for model checkpointing and early stopping.

**3. Transfer Learning:**

I. Loading the pre-trained model: To load a pre-trained CNN model with 'imagenet' weights we plan to use the VGG16 function from the Keras library.
II. Modifying top layers: We plan to add new layers on top of the pre-trained model for task-specific classification.
III. Freeze pre-trained layers: Set the trainable parameter of pre-trained layers to False to freeze their weights during training.
IV. Compile model: We will be specifying the optimizer (Adam), loss function (CategoricalCrossentropy), and evaluation metric (Accuracy).
V. Training the model: we will train only the top layers while keeping the pre-trained layers frozen.
VI. Fine-tune model: We will then Unfreeze pre-trained layers, set the trainable parameter to True, and re-compile and train the model with a lower learning rate to fine-tune the entire model.