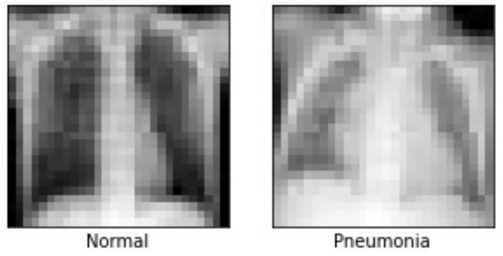
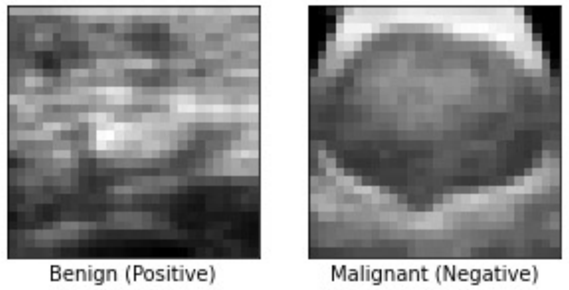
AutoKeras vs DARTS

An application of NAS to the BreastMNIST and PneumoniaMNIST Medical Datasets

The datasets

- **PneumoniaMNIST** is based on 5,856 pediatric chest X-ray images. The task is binary-class classification of pneumonia and normal. The source images are single-channel 1 × 28 × 28.
- **BreastMNIST** is based on a dataset of 780 breast ultrasound images The task is binary-class classification of malignant vs normal/benign. The source images are single-channel 1 × 28 × 28.

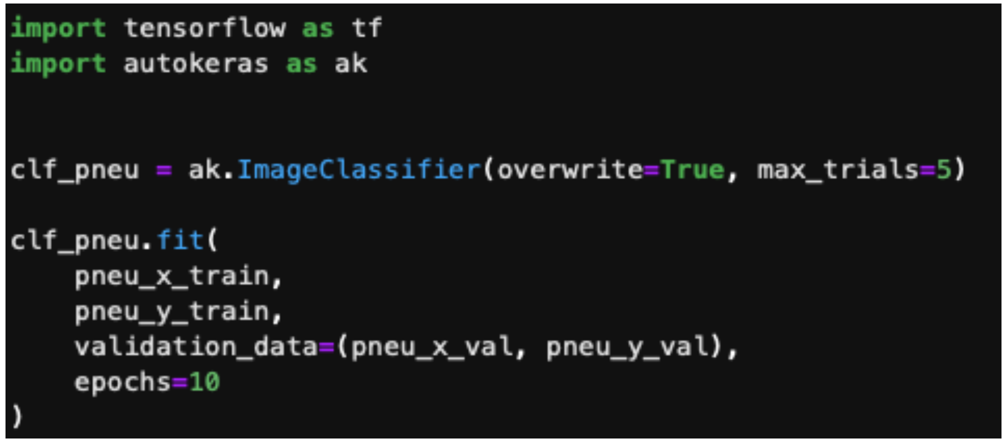


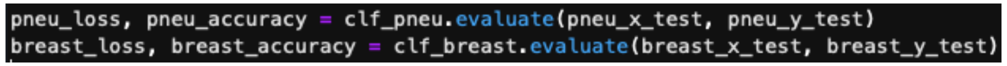


Neural Architecture Search

The problem: Creating neural networks for image datasets often involves challenges such as handling the complex, high-dimensional nature of image data, avoiding overfitting despite the large number of model parameters, requiring substantial computational resources for training and tuning, and needing significant expertise in model architecture design.

NAS: Neural Architecture Search (NAS) addresses some of these issues by automating the process of finding the optimal network architecture. It efficiently explores various configurations to balance model complexity and performance, thereby reducing the risk of overfitting. On the other hand, finding the best model can be computationally intensive

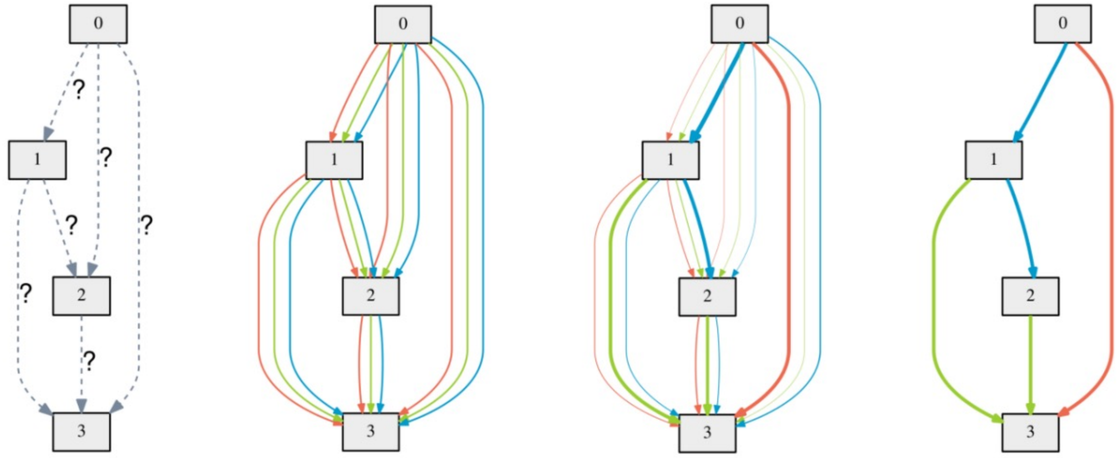

```
import tensorflow as tf
import autokeras as ak


clf_pneu = ak.ImageClassifier(overwrite=True, max_trials=5)

clf_pneu.fit(
    pneu_x_train,
    pneu_y_train,
    validation_data=(pneu_x_val, pneu_y_val),
    epochs=10
)
```


```
pneu_loss, pneu_accuracy = clf_pneu.evaluate(pneu_x_test, pneu_y_test)
breast_loss, breast_accuracy = clf_breast.evaluate(breast_x_test, breast_y_test)
```

AutoKeras

1. **Imports TensorFlow and AutoKeras**: TensorFlow is an open-source library for numerical computation and machine learning. AutoKeras is an AutoML library built on top of TensorFlow, designed to automate the process of model selection and hyperparameter tuning for machine learning models.
2. **Creates an Image Classifier**: Initializes an AutoKeras image classifier named clf_pneu to try a maximum of 5 different models (trials) to find the best one.
3. **Trains the Model**: The image classifier clf_pneu is trained using training data (pneu_x_train, pneu_y_train) and validation data (pneu_x_val, pneu_y_val) for a total of 10 epochs. This process involves the model learning from the training images and their labels, while also evaluating its performance on a separate set of validation images and labels to prevent overfitting.

DARTS

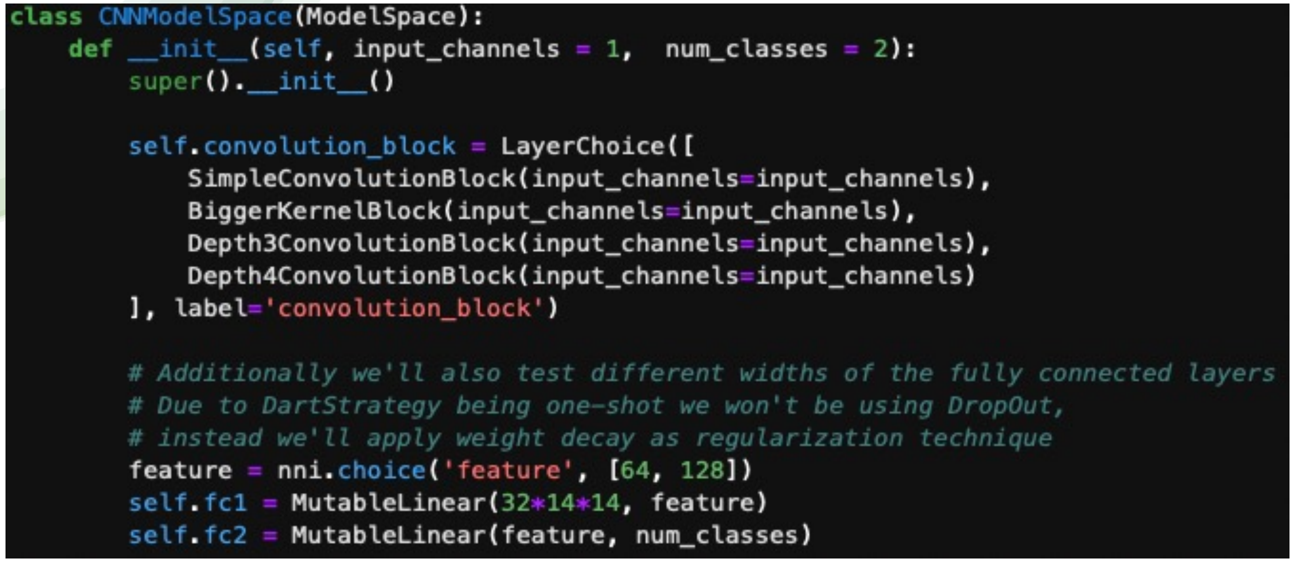


(a) (b) (c) (d)

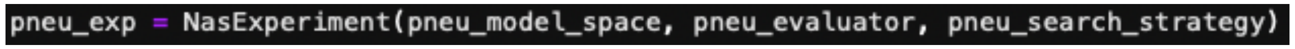
https://nni.readthedocs.io/en/latest/tutorials/darts.html

1. DARTS (Differentiable Architecture Search) introduces a gradient-based optimization approach. Unlike traditional NAS methods that rely on discrete and often computationally intensive search processes, DARTS transforms the architecture selection problem into a continuous one. It achieves this by relaxing the search space to be continuous, allowing the use of gradient descent for finding the optimal architecture.
2. DARTS is a one-shot strategy as it simultaneously evaluates a large number of architectural decisions using gradient descent without the need to train each architecture separately

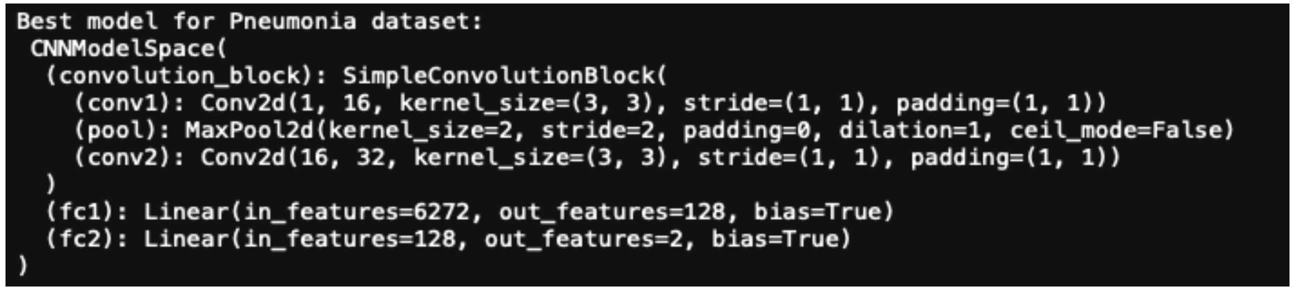
DARTS in Microsoft NNI



```
class CNNModelSpace(ModelSpace):
    def __init__(self, input_channels = 1,  num_classes = 2):
        super().__init__()

        self.convolution_block = LayerChoice([
            SimpleConvolutionBlock(input_channels=input_channels),
            BiggerKernelBlock(input_channels=input_channels),
            Depth3ConvolutionBlock(input_channels=input_channels),
            Depth4ConvolutionBlock(input_channels=input_channels)
        ], label='convolution_block')

        # Additionally we'll also test different widths of the fully connected layers
        # Due to DartStrategy being one-shot we won't be using DropOut,
        # instead we'll apply weight decay as regularization technique
        feature = nni.choice('feature', [64, 128])
        self.fc1 = MutableLinear(32*14*14, feature)
        self.fc2 = MutableLinear(feature, num_classes)
```



```
pneu_exp = NasExperiment(pneu_model_space, pneu_evaluator, pneu_search_strategy)
```



```
Best model for Pneumonia dataset:
 CNNModelSpace(
  (convolution_block): SimpleConvolutionBlock(
    (conv1): Conv2d(1, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv2): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  )
  (fc1): Linear(in_features=6272, out_features=128, bias=True)
  (fc2): Linear(in_features=128, out_features=2, bias=True)
)
```

Inputs:

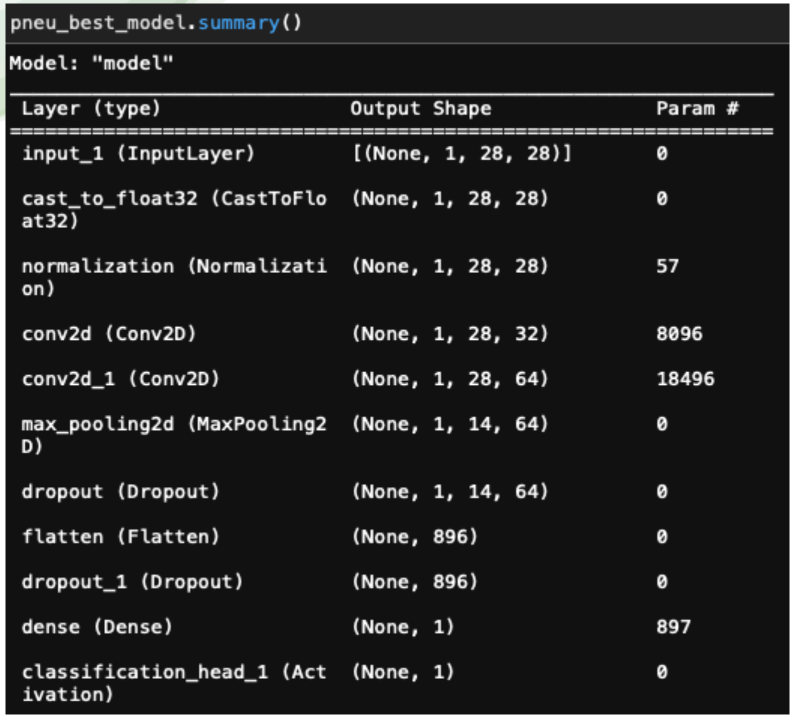
1. Define a Model Space with different architecture choices
2. Define a model evaluator
3. Define a model search strategy

Outputs:

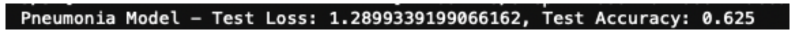
1. Best Model inside the model Space
2. Then we train using the best model

Results PneumoniaMNIST

AutoKeras

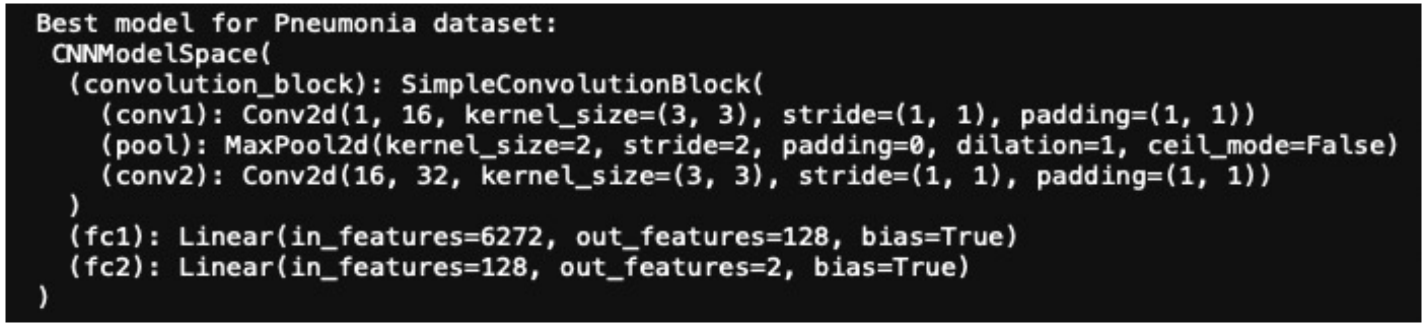


```
pneu_best_model.summary()

Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 1, 28, 28)]       0

 cast_to_float32 (CastToFlo  (None, 1, 28, 28)         0
 at32)

 normalization (Normalizati  (None, 1, 28, 28)         57
 on)

 conv2d (Conv2D)             (None, 1, 28, 32)         8096

 conv2d_1 (Conv2D)           (None, 1, 28, 64)         18496

 max_pooling2d (MaxPooling2  (None, 1, 14, 64)         0
 D)

 dropout (Dropout)           (None, 1, 14, 64)         0

 flatten (Flatten)           (None, 896)               0

 dropout_1 (Dropout)         (None, 896)               0

 dense (Dense)               (None, 1)                 897

 classification_head_1 (Act  (None, 1)                 0
 ivation)
```

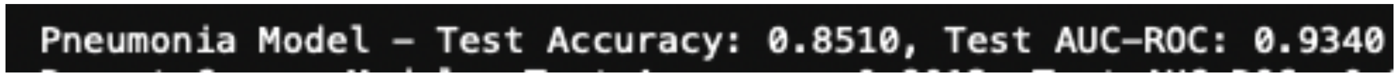


```
Pneumonia Model - Test Loss: 1.2899339199066162, Test Accuracy: 0.625
```

DARTS



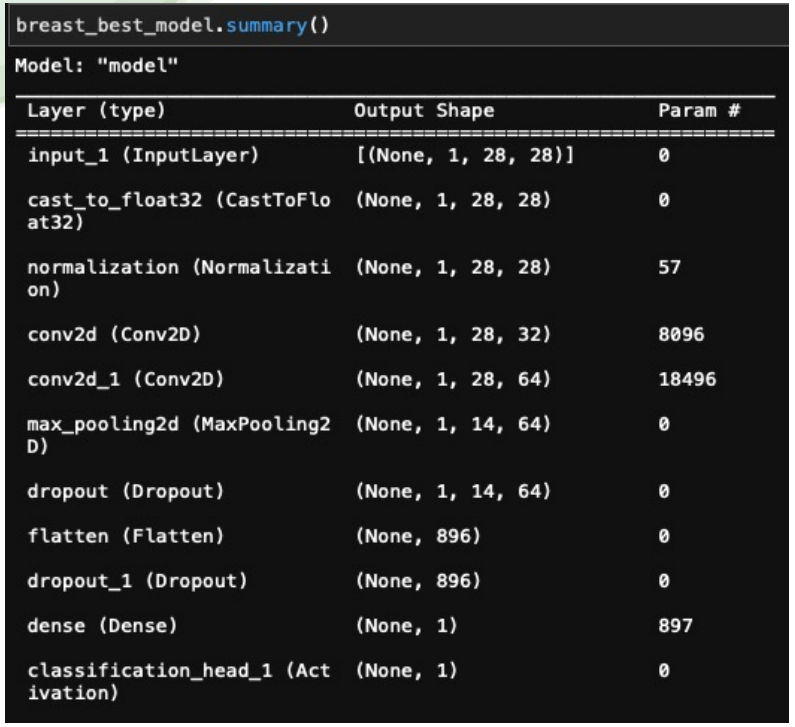
```
Best model for Pneumonia dataset:
 CNNModelSpace(
  (convolution_block): SimpleConvolutionBlock(
    (conv1): Conv2d(1, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv2): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  )
  (fc1): Linear(in_features=6272, out_features=128, bias=True)
  (fc2): Linear(in_features=128, out_features=2, bias=True)
)
```



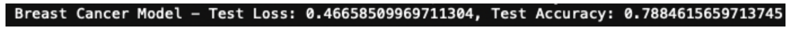
```
Pneumonia Model - Test Accuracy: 0.8510, Test AUC-ROC: 0.9340
```

Results BreastMNIST

AutoKeras

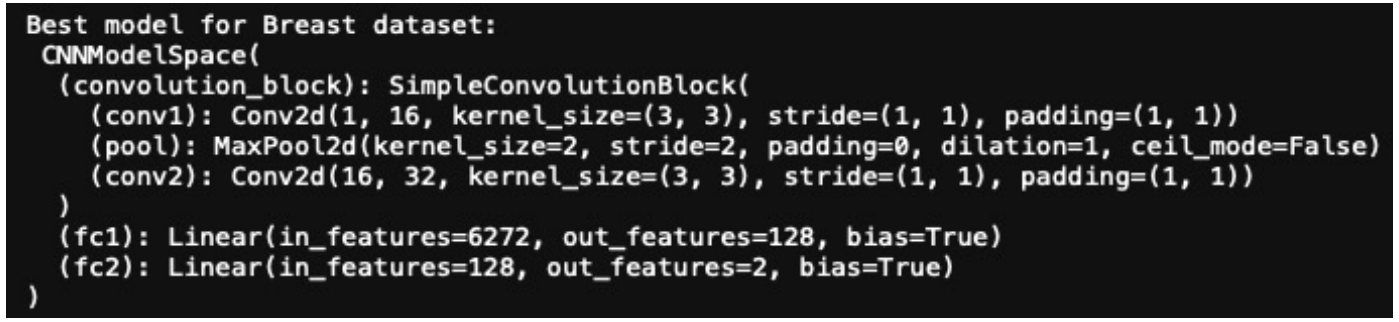


```
breast_best_model.summary()

Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 1, 28, 28)]       0

 cast_to_float32 (CastToFlo  (None, 1, 28, 28)         0
 at32)

 normalization (Normalizati  (None, 1, 28, 28)         57
 on)

 conv2d (Conv2D)             (None, 1, 28, 32)         8096

 conv2d_1 (Conv2D)           (None, 1, 28, 64)         18496

 max_pooling2d (MaxPooling2  (None, 1, 14, 64)         0
 D)

 dropout (Dropout)           (None, 1, 14, 64)         0

 flatten (Flatten)           (None, 896)               0

 dropout_1 (Dropout)         (None, 896)               0

 dense (Dense)               (None, 1)                 897

 classification_head_1 (Act  (None, 1)                 0
 ivation)
```

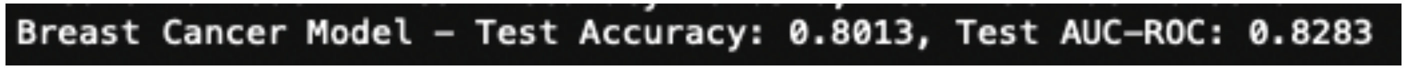


```
Breast Cancer Model - Test Loss: 0.466585099697113 04, Test Accuracy: 0.7884615659713745
```

DARTS



```
Best model for Breast dataset:
 CNNModelSpace(
  (convolution_block): SimpleConvolutionBlock(
    (conv1): Conv2d(1, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv2): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  )
  (fc1): Linear(in_features=6272, out_features=128, bias=True)
  (fc2): Linear(in_features=128, out_features=2, bias=True)
)
```



```
Breast Cancer Model - Test Accuracy: 0.8013, Test AUC-ROC: 0.8283
```

References

MedMinst: Yang, J., Shi, R., Wei, D. *et al.* MedMNIST v2 - A large-scale lightweight benchmark for 2D and 3D biomedical image classification. *Sci Data* **10**, 41 (2023). https://doi.org/10.1038/s41597-022-01721-8

Autokeras:
https://autokeras.com/tutorial/image_classification/

Microsoft NNI:
https://nni.readthedocs.io/en/latest/tutorials/darts.html
https://nni.readthedocs.io/en/latest/tutorials/hello_nas.html
https://nni.readthedocs.io/en/latest/nas/exploration_strategy.html#one-shot-nas
https://nni.readthedocs.io/en/latest/nas/evaluator.html#use-evaluators-to-train-and-evaluate-models