[1]:
```python
import h5py
import matplotlib.pyplot as plt
import numpy as np
import tensorflow.keras as K
from sklearn.model_selection import train_test_split
import pandas as pd
from sklearn.metrics import accuracy_score
from sklearn.utils import shuffle
```

1 Loading the Data

[2]:
```python
with h5py.File("info/data/output_signal.h5", "r") as file:
    signal_data = file["events"][:]

with h5py.File("info/data/output_bg.h5", "r") as file:
    bg_data = file["events"][:]
```

[3]:
```python
# storing signal and background data in panda DataFrame
signal = pd.DataFrame(signal_data)
background = pd.DataFrame(bg_data)

# concatenating the data frames to be part of one big set
df = pd.concat([signal, background])

# reseting the indicies
df = df.reset_index()

# creating the labels for the data sets i.e; signal = 1, background = 0 for␣
↪classification
labels = np.concatenate([np.ones(signal.shape[0]), np.zeros(background.
↪shape[0])])
labels = pd.DataFrame({'ttZ': labels})

# adding labels as a column at the end of the DataFrame
df = df.join(labels)

# shuffling the DataFrame
df_shuffled = df.sample(frac=1, random_state=42)  # 'random_state' for␣
↪reproducibility
df_shuffled.head()

# splitting the labels from the rest of the dataset
y = df_shuffled['ttZ']

# we can check the class distribution in the subset
print('ttZ events: {:.2f}%'.format(np.sum(y)/len(y) * 100))
```

```python
print('WZ events: {:.2f}%'.format((1 - np.sum(y)/len(y)) * 100))
```

```
ttZ events: 68.65%
WZ events: 31.35%
```

2 Various Feature Lists

[4]:
```python
# base input list suggested by the project
input_list = [ "H_T",
              "jet_1_pt",
              "jet_2_pt",
              "lep_1_pt",
              "lep_2_pt",
              "n_bjets",
              "jet_1_twb",
              "jet_2_twb",
              "bjet_1_pt"]

# best input list suggested by SRS
input_list_2 = ['jet_1_pt',
                'jet_3_pt',
                'jet_1_eta',
                'jet_3_eta',
                'jet_1_twb',
                'jet_2_twb',
                'jet_3_twb',
                'bjet_1_pt',
                'n_jets',
                'n_bjets',
                'n_leptons']

# smallest input list that still has an accuracy >80%, suggested by SRS
input_list_3 = ['jet_3_pt',
                'jet_1_twb',
                'jet_2_twb',
                'jet_3_twb',
                'bjet_1_pt',
                'n_jets',
                'n_bjets']

input_lists = [input_list, input_list_2, input_list_3]
```

3 Defining Model Function

[5]:
```
def build_model(lr=0.002, layer1=25, layer2=12, layer3=5):
    model = K.Sequential([
        preprocessing_layer,
        K.layers.Dense(layer1, activation="relu"),
        K.layers.Dense(layer2, activation="relu"),
        K.layers.Dense(layer3, activation="relu"),
        K.layers.Dense(1, activation="sigmoid")
    ])
    model.compile(optimizer=K.optimizers.Adam(learning_rate=lr),␣
 ↪loss='binary_crossentropy', metrics=['accuracy'])
    return model
```

4 Importing Best Results

[6]:
```
# from model_selection.ipynb
results = pd.read_csv('model_selection.csv')
results
```

[6]:

	Unnamed: 0	base list	SRS: best list	SRS: best smallest list
0	batch_size	300.0000	100.0000	150.0000
1	model__layer1	50.0000	25.0000	50.0000
2	model__layer2	12.0000	25.0000	50.0000
3	model__layer3	10.0000	15.0000	5.0000
4	model__optimizer__lr	0.0020	0.0020	0.0002
5	best score	0.8296	0.8339	0.8270

5 Feature Importance through Permutations

[7]:
```
# Function to calculate permutation feature importance
def permutation_importance(model, X_valid, y_valid, metric=accuracy_score):

    # Store the baseline accuracy of the model on original data
    baseline_accuracy = metric(y_valid, model.predict(X_valid).round())

    # accuracy decreases will be stored here
    importances = []

    # iterations over each feature
    for i in range(X_valid.shape[1]):
        save = X_valid[:, i].copy()

        # shuffle individual feature
        X_valid[:, i] = shuffle(X_valid[:, i])
        m_accuracy = metric(y_valid, model.predict(X_valid).round())
```

```python
        # restore original data
        X_valid[:, i] = save

        # store decrease in accuracy for feature i
        importances.append(baseline_accuracy - m_accuracy)

    return np.array(importances)
```

6 Running Best Models For Each Feature List And Ranking their Feature Importances

[8]:
```python
# fig for all best model performances from each feature list
fig, axs = plt.subplots(3, 3, figsize=(12, 16), sharey='row')

#
fig2, axs2 = plt.subplots(3, 1, figsize=(12, 16))

for i, feature_list in enumerate(input_lists):
    names = ['base list', 'SRS: best list', 'SRS: best smallest list']

    ## train, test, val splits
    X = df_shuffled[feature_list]
    x_train, x_rem, y_train, y_rem = train_test_split(X, y, train_size=0.8)
    x_val, x_test, y_val, y_test = train_test_split(x_rem, y_rem, train_size=0.5)

    ## building model
    preprocessing_layer = K.layers.Normalization()
    preprocessing_layer.adapt(x_train)

    model = build_model(lr=results[names[i]][4],
                        layer1=int(results[names[i]][1]),
                        layer2=int(results[names[i]][2]),
                        layer3=int(results[names[i]][3]))

    # fitting model
    fit_history = model.fit(
        x_train,
        y_train,
        batch_size=int(results[names[i]][0]),
        epochs=100,
        validation_data=(x_val, y_val),
        verbose = 0)


    print("Printing summary of the trained model:")
```

```
    print(model.summary())

    titles = ['Base List', 'SRS: Best List', 'SRS: Smallest List with >80% Accuracy']

    # plotting performance results
    axs[0, i].plot(fit_history.history["loss"], label="training")
    axs[0, i].plot(fit_history.history["val_loss"], label="validation")
    axs[0, i].legend()
    axs[0, i].set_title(titles[i])

    axs[1, i].plot(fit_history.history["accuracy"], label="training")
    axs[1, i].plot(fit_history.history["val_accuracy"], label="validation")
    axs[1, i].legend()

    _, bins, _ = axs[2, i].hist(model.predict(x_test[y_test.astype(bool)]), bins=20, alpha=0.3, density=True, label="test signal")
    axs[2, i].hist(model.predict(x_test[~y_test.astype(bool)]), bins=bins, alpha=0.3, density=True, label="test bg")
    axs[2, i].hist(model.predict(x_train[y_train.astype(bool)]), bins=bins, density=True, histtype="step", label="train signal")
    axs[2, i].hist(model.predict(x_train[~y_train.astype(bool)]), bins=bins, density=True, histtype="step", label="train bg")
    axs[2, i].legend()

    if i == 0:
            axs[0, i].set_ylabel("Loss")
            axs[1, i].set_ylabel("Accuracy")

    elif i == 1:
        axs[0, i].set_xlabel("Number of epochs")
        axs[1, i].set_xlabel("Number of epochs")
        axs[2, i].set_xlabel("NN output")


    # plotting feature rankings through the permutation method
    feature_importances = permutation_importance(model, x_test.to_numpy(), y_test.to_numpy())

    # Sorting the features by importance
    df = pd.DataFrame({'Label': x_test.columns, 'Value': feature_importances})
    df = df.sort_values('Value', ascending=False)

    # Plotting the sorted data
    axs2[i].bar(df['Label'], df['Value'], color='blue')
    axs2[i].set_xticklabels(df['Label'], rotation=45)
```

```python
    axs2[i].set_title(titles[i])

    if i == 1:
        axs2[i].set_ylabel('Decrease in Accuracy')
    elif i == 2:
        axs2[i].set_xlabel('Features')

    # evaluating test sets
    probs = model.predict(x_test)
    y_pred = (probs > 0.5).astype(int)
    print(names[i])
    print(accuracy_score(y_test, y_pred))

fig.tight_layout()
fig.savefig("best_models.pdf")

fig2.tight_layout()
fig2.savefig("feature_importances.pdf")

plt.show()
```

```
Printing summary of the trained model:
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 normalization (Normalizatio  (None, 9)                19
 n)

 dense (Dense)               (None, 50)                500

 dense_1 (Dense)             (None, 12)                612

 dense_2 (Dense)             (None, 10)                130

 dense_3 (Dense)             (None, 1)                 11

=================================================================
Total params: 1,272
Trainable params: 1,253
Non-trainable params: 19
_________________________________________________________________
None
1590/1590 [==============================] - 1s 371us/step
728/728 [==============================] - 0s 374us/step
12727/12727 [==============================] - 5s 409us/step
5812/5812 [==============================] - 2s 416us/step
2318/2318 [==============================] - 1s 447us/step
```

```
2318/2318 [==============================] - 1s 479us/step
2318/2318 [==============================] - 1s 453us/step
2318/2318 [==============================] - 1s 461us/step
2318/2318 [==============================] - 1s 441us/step
2318/2318 [==============================] - 1s 445us/step
2318/2318 [==============================] - 1s 527us/step
2318/2318 [==============================] - 1s 453us/step
2318/2318 [==============================] - 1s 491us/step
2318/2318 [==============================] - 1s 445us/step
 387/2318 [===>...] - ETA: 0s
```

```
/var/folders/3x/lv7sddxn2gg8mq0dwdcn1wg40000gn/T/ipykernel_79200/1070017733.py:7
5: UserWarning: set_ticklabels() should only be used with a fixed number of
ticks, i.e. after set_ticks() or using a FixedLocator.
  axs2[i].set_xticklabels(df['Label'], rotation=45)
```

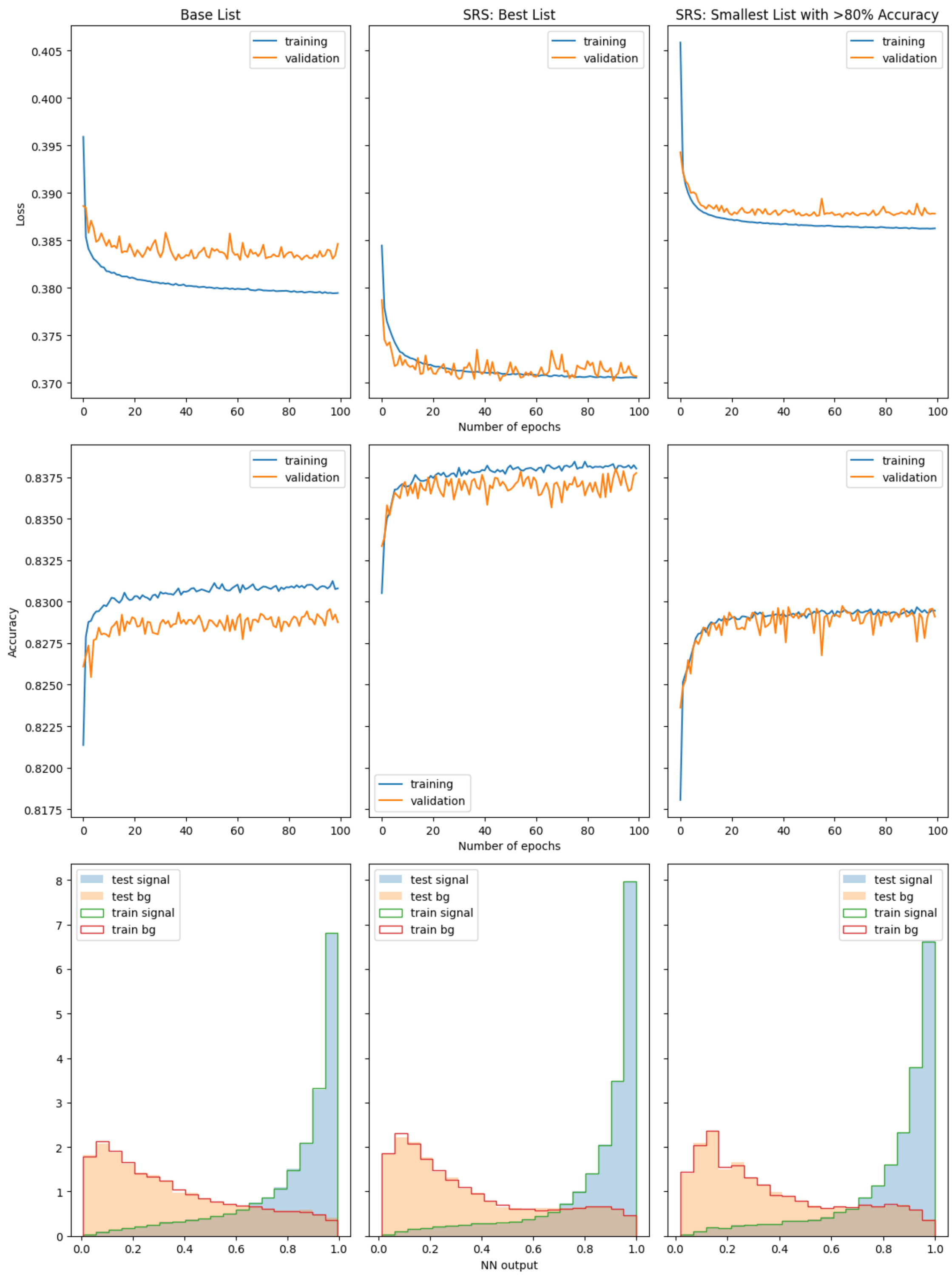
```
2318/2318 [==============================] - 1s 402us/step
base list
0.8290854168352347
Printing summary of the trained model:
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 normalization_1 (Normalizat  (None, 11)               23
 ion)

 dense_4 (Dense)             (None, 25)                300

 dense_5 (Dense)             (None, 25)                650

 dense_6 (Dense)             (None, 15)                390

 dense_7 (Dense)             (None, 1)                 16

=================================================================
Total params: 1,379
Trainable params: 1,356
Non-trainable params: 23
_________________________________________________________________
None
1589/1589 [==============================] - 1s 371us/step
729/729 [==============================] - 0s 370us/step
12725/12725 [==============================] - 5s 367us/step
5814/5814 [==============================] - 2s 386us/step
2318/2318 [==============================] - 1s 372us/step
2318/2318 [==============================] - 1s 377us/step
2318/2318 [==============================] - 1s 369us/step
2318/2318 [==============================] - 1s 367us/step
```

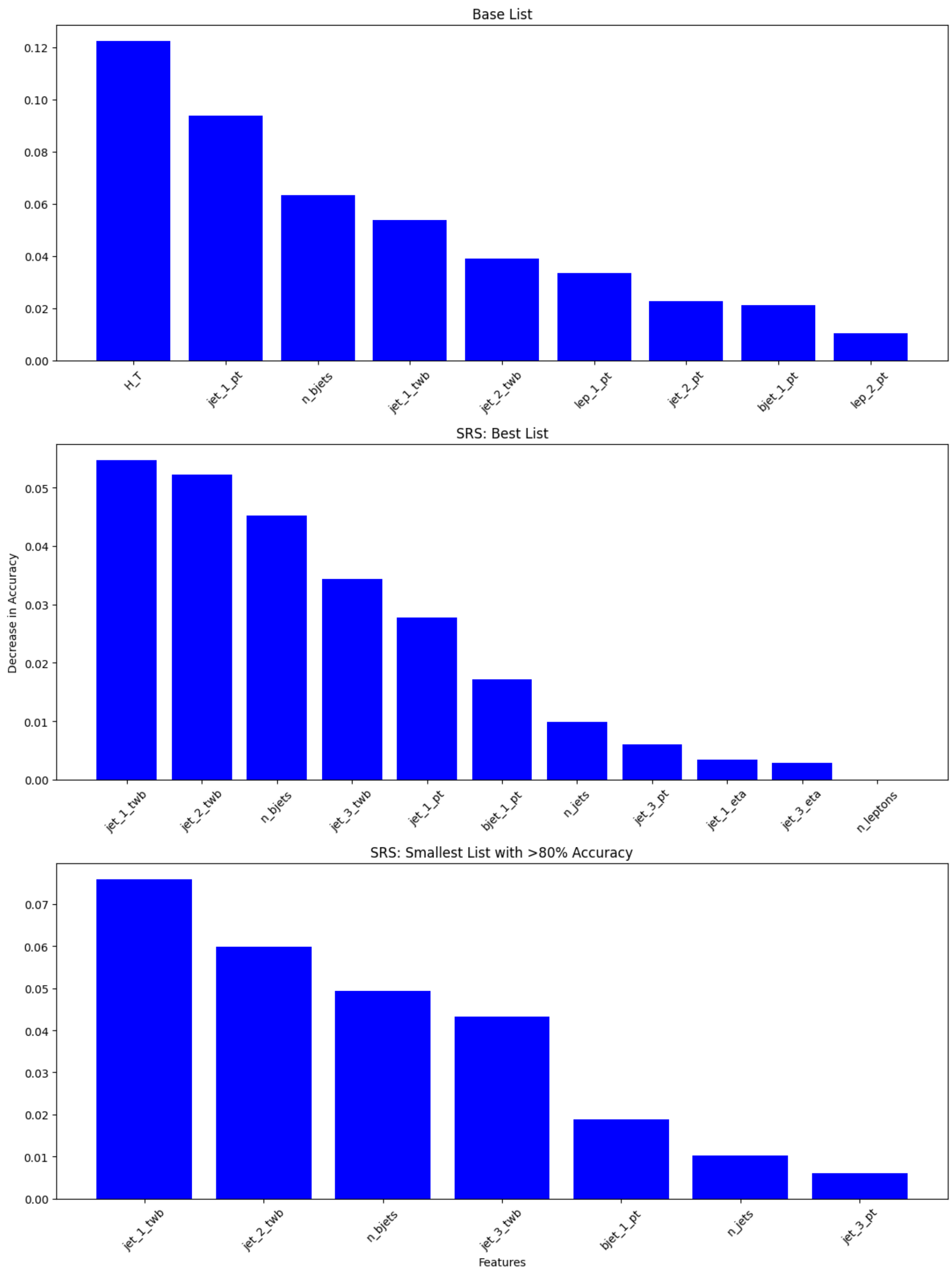
```
2318/2318 [==============================] - 1s 610us/step
2318/2318 [==============================] - 1s 376us/step
2318/2318 [==============================] - 1s 370us/step
2318/2318 [==============================] - 1s 369us/step
2318/2318 [==============================] - 1s 367us/step
2318/2318 [==============================] - 1s 367us/step
2318/2318 [==============================] - 1s 417us/step
2318/2318 [==============================] - 1s 409us/step
 406/2318 [====>...] - ETA: 0s
```

```
/var/folders/3x/lv7sddxn2gg8mq0dwdcn1wg40000gn/T/ipykernel_79200/1070017733.py:7
5: UserWarning: set_ticklabels() should only be used with a fixed number of
ticks, i.e. after set_ticks() or using a FixedLocator.
  axs2[i].set_xticklabels(df['Label'], rotation=45)
```

```
2318/2318 [==============================] - 1s 368us/step
SRS: best list
0.839482698168676
Printing summary of the trained model:
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 normalization_2 (Normalizat  (None, 7)                15
 ion)

 dense_8 (Dense)             (None, 50)                400

 dense_9 (Dense)             (None, 50)                2550

 dense_10 (Dense)            (None, 5)                 255

 dense_11 (Dense)            (None, 1)                 6

=================================================================
Total params: 3,226
Trainable params: 3,211
Non-trainable params: 15
_________________________________________________________________
None
1595/1595 [==============================] - 1s 440us/step
723/723 [==============================] - 0s 472us/step
12721/12721 [==============================] - 5s 385us/step
5818/5818 [==============================] - 2s 387us/step
2318/2318 [==============================] - 1s 417us/step
2318/2318 [==============================] - 1s 374us/step
2318/2318 [==============================] - 1s 377us/step
2318/2318 [==============================] - 1s 375us/step
2318/2318 [==============================] - 1s 376us/step
```

```
2318/2318 [==============================] - 1s 375us/step
2318/2318 [==============================] - 1s 375us/step
2318/2318 [==============================] - 1s 374us/step
 400/2318 [====>...] - ETA: 0s

/var/folders/3x/lv7sddxn2gg8mq0dwdcn1wg40000gn/T/ipykernel_79200/1070017733.py:7
5: UserWarning: set_ticklabels() should only be used with a fixed number of
ticks, i.e. after set_ticks() or using a FixedLocator.
  axs2[i].set_xticklabels(df['Label'], rotation=45)

2318/2318 [==============================] - 1s 378us/step
SRS: best smallest list
0.8313240014024867
```


Base List
SRS: Best List
SRS: Smallest List with >80% Accuracy
Loss
Accuracy
Number of epochs
NN output
training
validation
test signal
test bg
train signal
train bg


Base List
0.12
0.10
0.08
0.06
0.04
0.02
0.00
H_T
jet_1_pt
n_bjets
jet_1_twb
jet_2_twb
lep_1_pt
jet_2_pt
bjet_1_pt
lep_2_pt
SRS: Best List
Decrease in Accuracy
0.05
0.04
0.03
0.02
0.01
0.00
jet_1_twb
jet_2_twb
n_bjets
jet_3_twb
jet_1_pt
bjet_1_pt
n_jets
jet_3_pt
jet_1_eta
jet_3_eta
n_leptons
SRS: Smallest List with >80% Accuracy
0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.00
jet_1_twb
jet_2_twb
n_bjets
jet_3_twb
bjet_1_pt
n_jets
jet_3_pt
Features