Algorithm 1 **Analog-SNN Synthesis Framework with Deep Learning**

```
1. Load post-layout (PLS) transfer function of eNeuron
transfer_functions = load_PLSresults(model_eNeuron)
2. Activation Function Identification
activation_function = (
  transfer_functions,
  normalization
  fitting: polynomial_fit or sigmoid_fit )
3. Network Structure
Structure = [            ▷ Structure could be adjusted for any other problem
  inputs = Input(shape=(num_regions,))
  x = Dense(12, activation=activation_function)(inputs)
  x = Dense(12, activation=activation_function)(x)
  region_output = Dense(num_region_classes, activation=activation_function, name='region_output')(x)
  y = Concatenate()([x, region_output])
  angle_output = Dense(num_angle_classes, activation=activation_function, name='angle_output')(y)
  model = Model(inputs=inputs, outputs=[region_output, angle_output])
]
4. Network Model for Training and Testing on Simulated or Measured Dataset
model_training = (Structure, activation_function,train_data)
model_testing = (Structure, activation_function, test_data)
5. Network Model Training
for $epoch = 1$ to 100 do
   accuracy_training, tensor_weight= learning(model_training, epoch)
end for
5. Weight Extraction
if accuracy_training $\geq 0.98$ then
   trained_weight = tensor_weight
else if $epoch == 100$ then
   trained_weight = tensor_weight
end if
6. Transistor Variability consideration in Weight
$\bar{w}$ = trained_weight
$\sigma_w = 0.01^*$ trained_weight
statistic_weight = normal_distribution($\bar{w}$, $\sigma_w$)
7. Network Model Testing
accuracy_testing = testing(model_testing , statistic_weight)
```