Modifying The First Layer With 64 Units & The Activation Func. With Elu

- The first layer acts as the primary feature extractor. By giving it more units and a better gradient flow (ELU), the model can derive higher-quality features.
- **Increasing the number of units (from 32 to 64) in the first layer** : provides the model with more capacity to learn more complex patterns and features.
- **ELU** generally leads to better gradient flow, helping the model converge more effectively, especially in Complex Features.

Modifying The First Layer With 128 Units & All Layers With Elu Activation Func.

- Excessive units can lead to overfitting the model.
- If the input data doesn't match the high-dimensional units , the additional units might introduce noise or redundancy in the learned features, negatively impacting performance.
- Using ELU for shallow (مسطحة) layers (e.g., the first layer) can be helpful because these layers deal with raw input features.
- In deeper layers, where features are already more abstract, the ReLU activation's is more advantageous. Producing zero outputs in hidden layers aids to generalization.

Modifying The First Layer With 64 Units & Activation Func With Elu & Stddev With .06 Instead Of .05 (Custom Initializer)

- Initialization affects how quickly and effectively a model converges by controlling the initial of weights. If weights are too large, activations can explode; if too small, gradients can vanish.
- Suitable initialization balances the network's learning dynamics, allowing it toconverge better and faster.

Using Predefined Initializer (Glorot Normal) Instead Of Custom Initializer

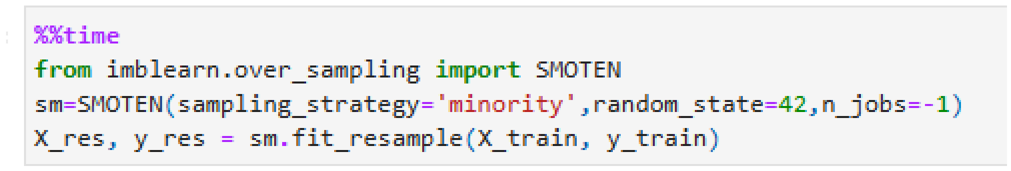
- **Its parameters :**
  1. **Mean** = 0
  2. **Standard Deviation** = $\sqrt{\frac{2}{\text{fan\_in}+\text{fan\_out}}}$

     fan_in: Number of input units to the layer.
     fan_out: Number of output units from the layer.

- Designed to maintain a balance between variance of activations and gradients.

1. **Dropout** reduces overfitting by promoting independence among neurons and limiting the model's ability to memorize the training data. Adjusting the dropout rate allows you to balance.

2. **Adam Optimizer** in a neural network determines how the model's weights are updated during training to minimize the loss function by calculating how the true labels far from the predicted labels.



```
%%time
from imblearn.over_sampling import SMOTEN
sm=SMOTEN(sampling_strategy='minority',random_state=42,n_jobs=-1)
X_res, y_res = sm.fit_resample(X_train, y_train)
```

3. Generates synthetic (اصطناعي) samples for the class by interpolating between existing samples in the feature space. It is specifically designed for categorical data which is unbalanced.
   - `'auto'` (default): Balances all classes equally.
   - `n_jobs=-1`: Utilizes all available CPU cores for parallel processing for speeding up.