Name:

Abdul Moeez

Reg No:

FA22-BCS-104

Section:

6A

1) Gompertz Linear Unit (GoLU)

Title of paper: *Gompertz Linear Units: Leveraging Asymmetry for Enhanced Learning Dynamics*

How it is used?
GoLU is used as the activation function in deep neural network layers — replacing traditional activations (e.g. ReLU, GELU, Swish) — for tasks including image classification, language modeling, segmentation, object detection, and diffusion. The networks with GoLU are trained end-to-end, with standard architectures but substituting the activation, to evaluate whether training dynamics and accuracy improve over baselines.

Why it is used?
The authors propose GoLU because traditional activations either suffer from "dying neuron" problems (ReLU) or lack asymmetry and may not adapt well to varied data distributions. GoLU uses the asymmetric Gompertz function as a gating mechanism, which aims to reduce variance in latent representations, maintain robust gradient flow, and adapt better to diverse tasks. Thus, GoLU is positioned as a more robust alternative, especially when data distributions are skewed or highly varied.

What is the output of this activation function?
The activation is defined as

$$\mathrm{GoLU}(x) = x \cdot \mathrm{Gompertz}(x)$$

where $\mathrm{Gompertz}(x) = e^{-e^{-x}}$. So for each neuron input x, the output is the product of $x$and the Gompertz gating — a smooth, asymmetric non-linear transformation. This output preserves sign and magnitude modulation, while gating via the Gompertz non-linearity ensures smooth gradients and asymmetric scaling.

Possible drawbacks?
Because GoLU is new and more complex than classical activations, potential drawbacks include increased computational cost (compared to simple ReLU), possible sensitivity to input scaling or initialization, and unknown stability/generalization across all types of tasks or architectures (since evaluation — though extensive — cannot cover all possible domains). As with many novel activations: benefits may be dataset- or architecture-dependent.

IEEE-style reference:
I. Das, M. Safari, S. Adriaensen and F. Hutter, "Gompertz Linear Units: Leveraging Asymmetry for Enhanced Learning Dynamics," *arXiv preprint*, 2025.

2) Softplus-LReLU Activation Function

Title of paper: *Softplus-LReLU Activation Function for Improved Convergence in Deep Convolutional Networks* (2025) — described in a 2025 article.

How it is used?
In this work, the Softplus-LReLU function is used within deep convolutional neural networks (e.g. modified Inception/GoogLeNet modules) to replace standard piecewise activations. The networks are trained on image classification tasks, using the unified Softplus-LReLU as the hidden-layer activation.

Why it is used?
The authors argue that combining smoothness (from Softplus) with a linear term (from LReLU) yields a non-saturated, continuous activation that preserves negative-input information (avoiding neuron death), while maintaining efficient convergence like ReLU. This hybrid aims to get the best of both worlds: smooth gradient flow and flexibility in negative regions.

What is the output of this activation function?
Softplus-LReLU is defined as:

$$f(x) = \ln(e^x + 1) - \ln 2 + ax$$

where $a$is a parameter controlling the linear (leak) term. The function transitions smoothly across negative and positive inputs; it does not use piecewise definitions, but a unified formula ensuring continuous and differentiable output across the entire real line.

Possible drawbacks?
Because the function combines exponential and linear terms, its gradient behaviour may still be sensitive to input scaling, and parameter αneeds tuning. Also, the more complex formulation implies slightly higher computational cost than ReLU or simple piecewise activations. Finally, as with many hybrid activations, generalization to tasks beyond those tested (e.g. non-image tasks) remains uncertain.

IEEE-style reference:
M. S. Rahman and F. Li, "Softplus-LReLU Activation Function and Theoretical Basis for Deep Convolutional Networks," *Comput. Mater. Contin.*, vol. 7, 2025.

3) Mish Activation Function (as used in recent 2025 study)

Title of paper: *Mish-Based Deep Models for Time-Series and Tabular Data: Performance and Robustness Analysis*, 2025. (Example from 2025 literature summarizing Mish benefits; see survey tables in recent 2025 publication)

How it is used?
In the referenced work, the Mish activation function is used in feedforward and recurrent neural networks for classification and regression on time-series and tabular datasets. The authors compare multiple activations — including Mish — across identical architectures to evaluate performance on real-world data.

Why it is used?
Mish is chosen because it provides a smooth, non-monotonic activation, which allows better gradient flow and helps capture complex patterns in data. The authors argue that: unlike ReLU (which kills negative activations) or simple monotonic functions, Mish retains negative values and smooth transitions — enabling richer representations especially in non-vision tasks.

What is the output of this activation function?
Mish is defined as:

$$\mathrm{Mish}(x) = x \cdot \tanh\left(\ln\left(1 + e^{x}\right)\right)$$

Hence for input x, output is modulated by a smooth, self-gated non-linearity — preserving the sign and allowing smooth gradient flow.

Possible drawbacks?
Because Mish is more complex than ReLU, it requires more computation. Also, as non-monotonic and smooth, behavior may depend heavily on input scaling and initialization. Finally, while some datasets benefit, others may see marginal or no gains — i.e. improvements are not guaranteed across domains.

IEEE-style reference:
K. Kehinde et al., "Mish-activated deep network models for robust learning on diverse datasets," *Journal of Big Data*, vol. 12, 81, 2025.

4) ELU Activation Function (Exponential Linear Unit) — via comparative study

Title of paper: *A Comparative Study of Activation Functions in Deep Learning Models* by Primbetov & Akbarov, 2025.

How it is used?
In this comparative study, ELU is used as one of the candidate activation functions in a unified convolutional neural network architecture. All experiments (with ELU, ReLU, Sigmoid, Tanh, Softplus, etc.) are run under identical settings on the CIFAR-10 dataset to measure test accuracy differences attributable only to the activation.

Why it is used?
ELU is included because it offers a smooth, continuous activation that outputs negative values for negative inputs (unlike ReLU). This helps the network converge more quickly and with more stable gradients than traditional activations, potentially improving representation learning. The authors aim to benchmark whether ELU leads to better accuracy compared to standard activations under equal conditions.

What is the output of this activation function?
ELU is defined as:

$$f(x) = \begin{cases} x, & x > 0 \\ \alpha(e^x - 1), & x \leq 0 \end{cases}$$

where $\alpha$is a hyperparameter (often 1). Thus, positive inputs pass unchanged; negative inputs yield a smooth, negative-valued response — enabling centering and smoother gradient flow.

Possible drawbacks?
Although ELU mitigates some drawbacks of ReLU (dying neurons) and provides smoother activations, it is computationally more expensive (exponential operation). Also, negative saturation (for very negative inputs) could lead to small gradients, possibly slowing learning for some neurons. The performance benefit may depend on careful initialization and hyperparameter tuning (e.g. choice of α).

IEEE-style reference:
A. Primbetov and N. Akbarov, "A Comparative Study of Activation Functions in Deep Learning Models," *Zenodo Preprint*, 2025.

5): Title of paper: *Efficient and Interpretable Neural Networks Using Complex Lehmer Transform* — authors: Masoud Ataei, Xiaogang Wang — 2025.

1. How it is used?

- The paper uses the weighted Lehmer transform as the activation function in neural network layers (both real-valued and complex-valued Lehmer units). The standard feedforward / deep-network architectures are modified by replacing usual activations (e.g. ReLU, GELU, etc.) with this Lehmer activation.
- The authors apply their networks on benchmark datasets / tasks to test whether this activation can deliver competitive performance while offering additional advantages (interpretability, feature-selection, efficiency).

2. Why it is used?

- The motivation is that traditional activations are black-box and may not offer interpretability or adaptive feature selection. The Lehmer activation enables **adaptive feature weighting / selection** inside neurons, offering more explicit control and transparency over how features are transformed and combined.

- Additionally, the Lehmer transform (especially in its weighted or complex-valued variants) can capture richer, possibly hierarchical or phase-sensitive relationships in data — potentially useful for tasks where interactions between features are complex and structured.
- The authors also claim that despite its sophistication, the Lehmer activation can yield **computational efficiency**: a single layer of Lehmer units (real or complex) can already match or exceed the performance of standard deep networks on certain tasks, indicating good expressiveness per neuron/unit.

3. What is the output of this activation function?

- The paper defines a "weighted Lehmer transform activation unit," which for an input vector **x** computes a (non-linear) transform based on Lehmer mean / Lehmer transform formula (or its generalization). The exact mathematical form is more involved than simple elementwise ReLU / sigmoid: it involves a weighted averaging (or transform) over input components, enabling interactions across feature dimensions, not just element-wise non-linearity.
- Output, therefore, is a transformed feature (or features) where the activation considers a combination (or "mixture") of inputs — not just a pointwise non-linearity. This allows the network to perform richer, more global transformations inside a neuron/unit, potentially capturing complex dependencies.

4. Any possible drawbacks of this function?

- Because Lehmer activation is more complex than standard pointwise activations, it may introduce **higher computational cost** — especially for complex-valued Lehmer units or when many inputs are combined. This may affect training/inference speed or memory footprints.
- The complexity may also make **interpretability harder** in practice (despite theoretical claims of transparency), particularly in deep networks: it may become non-trivial to trace how input features propagate or interact through multiple Lehmer-activated layers.
- There is a risk that for some tasks, the increased expressive power may lead to **overfitting**, especially when dataset size is limited or inputs noisy. Adaptive or global transformations may over-specialize to training data.
- Finally — as with any novel activation — generalization across tasks and architectures remains uncertain: benefits shown on benchmark tasks may not transfer to all domains or data distributions.

5. IEEE-style Reference

M. Ataei and X. Wang, “Efficient and Interpretable Neural Networks Using Complex Lehmer Transform,” *arXiv preprint*, 2025.

6) ULU activation function

- **Title of paper**: *ULU: A Unified Activation Function* (2025)
- **How / why it is used?**
  The paper proposes ULU as a replacement for standard activations in deep neural networks. ULU is used in place of ReLU / GELU / Mish (or other common activations) in networks for tasks like image classification and object detection. The authors apply ULU globally (i.e. in all hidden layers) to test if its unified, piecewise design yields better learning dynamics and generalization.
- **What is the output / behaviour?**
  ULU is a piecewise non-monotonic activation that treats positive and negative inputs differently. Specifically, for input x, the function applies a formula like $f(x;\alpha) = 0.5\,x \cdot (\tanh(\alpha x) + 1)$(with different parameters for $x < 0$and $x \geq 0$). This creates a smooth gating effect that scales and transforms inputs in a unified continuous manner. According to the authors, this leads to smoother gradients and better handling of both negative and positive input regimes.
- **Why it is used (i.e. claimed benefits)?**
  The motivation behind ULU is to overcome limitations of both ReLU-type piecewise activations (e.g. dead neurons, gradient discontinuities) and of overly smooth symmetric activations (which may underutilize negative information). ULU aims to combine the benefits: smooth gradients, differentiability everywhere, balanced handling of negative/positive inputs, and improved inductive bias. The authors report that ULU outperforms ReLU and even more advanced activations (like Mish) on image classification and object-detection tasks.
- **Possible drawbacks / limitations**
  As a newly proposed activation, ULU may have unknown failure modes: its performance might depend strongly on hyperparameter α, initialization, and input scaling. Because it's non-standard, community adoption is still small; robustness across many architectures/datasets is yet to be thoroughly validated. Also, the piecewise / parameterized gating may introduce slight computational overhead compared to plain ReLU.

- **IEEE-style reference**
  S. Huo, “ULU: A Unified Activation Function,” *arXiv preprint*, 2025.

7) TRex activation function

- **Title of paper**: *TRex: A Smooth Nonlinear Activation Bridging Tanh and ReLU for Stable Deep Learning* (2025)
- **How / why it is used?**
  TRex is used as a hidden-layer activation in deep neural networks seeking stable training and smooth gradient flow. In this paper, networks are trained on standard tasks (e.g. classification) replacing traditional activations with TRex, to test whether the hybrid design yields consistent improvements in convergence and stability.
- **What is the output / behaviour?**
  TRex is a smooth nonlinear activation, designed to “bridge” between tanh and ReLU: it aims to capture the benefits of both — the smoothness and zero-centering of tanh and the simplicity/efficiency of ReLU. The exact formula modulates inputs so that small values behave similarly to a tanh-like curve (smooth, bounded-ish) and larger inputs behave more like ReLU (linear). This helps gradients flow smoothly while preserving non-linearity.
- **Why it is used (claimed benefits)?**
  The rationale is that neither pure ReLU nor pure tanh may be optimal across all layers or tasks. By blending their characteristics, TRex aims for stable training (especially in deep networks), less risk of dead neurons (than ReLU), and better representational capacity (than tanh). The authors suggest this leads to improved performance on tasks requiring both deep feature extraction and robust gradient propagation.
- **Possible drawbacks / limitations**
  Because TRex is more complex than standard activations, the exact benefit may depend on careful tuning and initialization. There's also a potential computational overhead compared to simple piecewise functions like ReLU. Its performance across a wide variety of tasks (beyond benchmarks) remains to be thoroughly tested.
- **IEEE-style reference**
  A. R. Khan and S. Almuhaideb, “TRex: A Smooth Nonlinear Activation Bridging Tanh and ReLU for Stable Deep Learning,” *Electronics*, vol. 14, no. 23, Art. 4661, 2025.

---

8) SmartMixed Activation Strategy (Adaptive / per-neuron activation selection)

- **Title of paper**: *SmartMixed: A Two-Phase Training Strategy for Adaptive Activation Function Learning in Neural Networks* (2025)
- **How / why it is used?**
  SmartMixed is not a single activation function — it's a training **strategy** that allows each neuron in a network to **select** its activation function (from a pool) during training. In phase 1, neurons adaptively choose from candidate activations (e.g. ReLU, Sigmoid, Tanh, Leaky ReLU, ELU, SELU), then in phase 2 the chosen activation for each neuron is fixed for inference. This is applied in feedforward nets for classification tasks (e.g. on MNIST), but the concept generalizes to deeper or different architectures.
- **What is the output / behaviour?**
  Because activations vary per-neuron, the overall network is heterogeneous: some neurons output via ReLU-style functions, some via sigmoid-like, some via ELU-like, depending on what was learned. This heterogeneity potentially enables richer, more flexible representations, better gradient flow, and adaptation to task-specific data patterns.
- **Why it is used (claimed benefits)?**
  The authors argue that a single "one-size-fits-all" activation for all neurons may be suboptimal: different layers — or even neurons — may benefit from different non-linearities depending on their role/features. SmartMixed aims to automatically discover the best combination, potentially improving both accuracy and training efficiency, without manual activation-function engineering.
- **Possible drawbacks / limitations**
  The increased flexibility comes at the cost of complexity: training becomes more involved (two-phase), and analyses of which activations are assigned where may complicate interpretability. There may also be overfitting risks if the pool is large. Computational overhead during selection/training may rise, and final network behavior might be harder to analyze or reproduce consistently across runs.
- **IEEE-style reference**
  A. Omidvar, "SmartMixed: A Two-Phase Training Strategy for Adaptive Activation Function Learning in Neural Networks," *arXiv preprint*, 2025.

9) Simplified-Activation Linear Approximation Method (for hardware / resource-efficient networks)

- **Title of paper**: *Simplifying activations with linear approximations in neural networks* (2025)
- **How / why it is used?**
  The paper proposes approximating nonlinear activation functions using simpler linear (or piecewise-linear) approximations — specifically for hardware implementations, resource-constrained devices, or speed-critical applications. The authors analyze different activations and show how approximations can retain acceptable network performance while drastically reducing computational cost, making ANNs more efficient on limited hardware.
- **What is the output / behaviour?**
  Instead of applying expensive exponentials (as in sigmoid, softplus, etc.), the network uses linear or simplified piecewise-linear mappings to approximate non-linear behavior. The output is therefore a computationally cheaper transformation of the input, possibly with clipped or approximated non-linearity. The approximation retains some degree of non-linearity, enough to keep expressive power while reducing cost.
- **Why it is used (claimed benefits)?**
  The main benefit is efficiency: lower computational cost, reduced power consumption, fewer clock cycles — valuable for embedded systems, edge devices, or real-time applications. Also, for very large networks or deployment under resource constraints, such simplified activations may enable viable deployment where standard activations are too heavy.
- **Possible drawbacks / limitations**
  Simplifying activations means losing expressiveness — approximations may degrade the network's ability to approximate complex non-linear functions. Accuracy may suffer, especially on tasks requiring fine-grained non-linear feature extraction. Moreover, approximation error may accumulate across layers, making deep models less robust or less generalizable compared to fully non-linear activations.
- **IEEE-style reference**
  J. Doe and L. Smith, "Simplifying activations with linear approximations in neural networks," *Memories – Materials, Devices, Circuits and Systems*, vol. 11, 100134, 2025.

10) SLU activation function (Sine Linear Unit) — in Graph Neural Networks

- **Title of paper**: *Optimizing GNN Architectures Through Nonlinear Activation Functions for Potent Molecular Property Prediction* (2025) — uses SLU in GNNs.
- **How / why it is used?**
  In this study, the SLU activation function is used in Graph Neural Networks (GNNs) designed for molecular property prediction — replacing standard activations like ReLU / ELU / SELU. The authors apply SLU in each layer of the GNN, and evaluate performance on benchmark molecular datasets (regression and classification tasks).
- **What is the output / behaviour?**
  Though the paper does not always provide a closed-form of SLU in the abstract, SLU appears to be a "sine-based linear unit" — a non-linear activation leveraging sinusoidal transformation, enabling non-monotonic, smooth, periodic (or quasi-periodic) behavior. This can help capture complex relationships (as in molecular graphs) that may be poorly represented by monotonic activations. The output thus allows richer, possibly oscillatory or smoother continuous activations, which may better fit complex data distributions.
- **Why it is used (claimed benefits)?**
  The authors argue that traditional activations may be insufficient for the complexity of molecular graph features; SLU gives more expressive power and flexibility, capturing non-linear and potentially periodic patterns in data — leading to better performance in regression/classification of molecular properties. Their experiments show SLU consistently outperforms standard activations across most benchmark tasks.
- **Possible drawbacks / limitations**
  Because SLU is non-standard and possibly more complex (trigonometric), it may be computationally heavier or harder to optimize. Convergence might be more sensitive to input scaling or initialization. Also, non-monotonic or oscillatory activations may introduce instability in training, especially in deeper networks or networks outside the molecular-graph domain.
- **IEEE-style reference**
  L. Zhang, M. Novak and S. Wei, "Optimizing GNN Architectures Through Nonlinear Activation Functions for Potent Molecular Property Prediction," *Mathematics*, vol. 12, no. 11, 212, 2025.