SmartMixed: A Two-Phase Training Strategy for Adaptive Activation Function Learning in Neural Networks

Amin Omidvar
Independent Researcher
Toronto, Canada
omidvaramin@gmail.com

***Abstract*—The choice of activation function plays a critical role in neural networks, yet most architectures still rely on fixed, uniform activation functions across all neurons. We introduce SmartMixed, a novel two-phase training strategy that allows networks to learn optimal per-neuron activation functions while preserving computational efficiency at inference. In the first phase, neurons adaptively select from a pool of candidate activation functions (ReLU, Sigmoid, Tanh, Leaky ReLU, ELU, SELU) using a differentiable hard mixture mechanism. In the second phase, each neuron's activation function is fixed according to the learned selection, resulting in a computationally efficient network that supports continued training with optimized vectorized operations. We evaluate SmartMixed on the MNIST dataset using feedforward neural networks of varying depths. Our analysis reveals that neurons in different layers exhibit distinct preferences for activation functions, providing insights into the functional diversity within neural architectures.**

***Index Terms*—neural networks, adaptive activation functions, machine learning, deep learning**

I. Introduction

The choice of activation function plays a crucial role in the performance of neural networks. However, most existing architectures assign a uniform activation function across all neurons within a layer, typically using ReLU, Sigmoid, or Tanh. While this approach simplifies implementation and supports efficient vectorized operations, it restricts neurons to a fixed activation function regardless of whether it is optimal for their position within the network. In this work, we allow each neuron to adaptively select its own activation function and analyze these behaviors to gain deeper insights into their preferences.

Traditional approaches typically assign a single activation function to all neurons in a network, treating it as a hyperparameter to be tuned for the entire model on a given dataset. While this design simplifies implementation, it limits flexibility by preventing individual neurons from adapting their activation functions independently. To address this limitation, recent methods have introduced ensembles of activation functions, such as weighted averages, to provide greater adaptability. Although these approaches allow neurons to express preferences over multiple activation functions, they often introduce substantial computational overhead, such as additional learnable parameters, the need to calculate multiple activation functions per neuron instead of just one, etc.

Code available at: https://github.com/omidvaramin/SmartMixed

We introduce SmartMixed, a novel two-phase training paradigm that enables each neuron to select its preferred activation function. In the first phase, the network is given full flexibility: each neuron learns which activation function best suits its role during training. In the second phase, these choices are fixed, and the entire network is further trained using the selected activation functions. Fixing the activations reduces computational cost, as only one activation function needs to be calculated per neuron, and it eliminates the additional learnable parameters required for activation selection.

The SmartMixed training strategy operates in two distinct phases:

- **Selection Phase**: All neurons, except those in the output layer, learn to select their optimal activation functions from a predefined pool, enabling differentiable discrete choices while preserving gradient flow.
- **Fixed Phase**: Based on the learned selections, we construct a computationally efficient network where each neuron uses its individually selected activation function, allowing continued training with standard vectorized operations.

II. Related Work

The problem of selecting optimal activation functions has been explored through several approaches. Traditional methods typically assign a single activation function globally across the entire network [1], with ReLU [2] and its variants like Leaky ReLU [3], ELU [4], and SELU [5] being widely adopted choices.

Recent work has focused on learning adaptive activation functions. Ramachandran et al. [6] used neural architecture search to discover new activation functions, but applied them uniformly across networks. Several approaches have explored trainable activation functions with learnable parameters [7], including parametric ReLU variants and sigmoid-weighted linear units [8]. However, these methods typically learn global parameters that affect all neurons similarly.

Mixture-based approaches have attempted to combine multiple activation functions. Some works use weighted combinations of activation functions [9] or dynamic selection mech-

anisms [10]. However, these approaches introduce significant computational overhead by requiring evaluation of multiple activation functions per neuron and additional learnable parameters for mixing weights.

Neural architecture search methods [11] have explored activation function selection as part of broader architectural optimization, but typically focus on layer-wise or block-wise choices rather than individual neuron preferences. Adaptive neural networks [12] have investigated dynamic architectures, but primarily focus on structural changes rather than activation function adaptation.

Unlike previous approaches, SmartMixed enables each individual neuron to learn and commit to its own optimal activation function from a predefined pool. After the initial learning phase, the resulting network operates with computational efficiency, while maintaining the benefits of per-neuron activation specialization. This fine-grained, neuron-level activation adaptation distinguishes our work from existing methods that operate at network or layer granularity.

III. Methodology

A. Problem Formulation

Consider a neural network with layers $l \in 1, 2, \ldots, L$, where layer l contains n_l neurons. In conventional neural networks, all neurons typically share a single activation function σ, chosen as a global hyperparameter (e.g., ReLU, Sigmoid, or Tanh), with the output layer often being an exception. In contrast, our approach aims to learn an individualized activation function $\sigma_{l,i}$ for each neuron i in layer l, selected from a predefined pool $\mathcal{A} = \{\text{ReLU}, \text{Sigmoid}, \text{Tanh}, \text{LeakyReLU}, \text{ELU}, \text{SELU}\}$.

B. Phase 1: Selective Activation Learning

In the selective phase, each neuron learns to choose its optimal activation function from a pool $\mathcal{A}$ using a differentiable discrete sampling mechanism. For neuron i in layer l, we maintain a learnable logit vector $\boldsymbol{\alpha}_{l,i} \in \mathbb{R}^{|\mathcal{A}|}$ that encodes its preference over the candidate activation functions.

To enable stochastic selection, we employ the hard Gumbel–Softmax estimator [13]. First, we draw a vector of independent and identically distributed (i.i.d.) uniform random variables and transform them into Gumbel$(0, 1)$ noise using the inverse cumulative distribution function:

$$\mathbf{v}_{l,i} \sim \text{Uniform}(0,1)^{|\mathcal{A}|}, \qquad \mathbf{g}_{l,i} = -\log\Big(-\log(\mathbf{v}_{l,i}+\varepsilon)+\varepsilon\Big), \tag{1}$$

where ε is a small constant added for numerical stability.

The noise-perturbed logits are normalized via the softmax with temperature τ:

$$y_{l,i,j} = \frac{\exp((\alpha_{l,i,j} + g_{l,i,j})/\tau)}{\sum_{k=1}^{|\mathcal{A}|} \exp((\alpha_{l,i,k} + g_{l,i,k})/\tau)}, \qquad j = 1, \ldots, |\mathcal{A}|. \tag{2}$$

To obtain a discrete selection while retaining differentiability during training, we adopt the straight-through (ST) Gumbel–Softmax estimator proposed by [13]:

$$\mathbf{h}_{l,i} = \text{one\_hot}\big(\arg\max(\mathbf{y}_{l,i})\big) + \big(\mathbf{y}_{l,i} - \text{sg}(\mathbf{y}_{l,i})\big), \tag{3}$$

where $\text{sg}(\cdot)$ is the stop-gradient operator. In the forward pass, $\text{sg}(\mathbf{y}_{l,i}) = \mathbf{y}_{l,i}$ in value, so $\mathbf{h}_{l,i}$ reduces to a hard one-hot vector. In the backward pass, the stop-gradient enforces $\frac{\partial\, \text{sg}(\mathbf{y}_{l,i})}{\partial \mathbf{y}_{l,i}} = 0$, so that

$$\frac{\partial \mathbf{h}_{l,i}}{\partial \mathbf{y}_{l,i}} = I. \tag{4}$$

Here the one-hot term contributes no gradient, since $\arg\max$ is discrete and non-differentiable. As a result, the gradients propagate as if $\mathbf{h}_{l,i} = \mathbf{y}_{l,i}$, ensuring smooth optimization while retaining hard one-hot behavior in the forward pass.

The output of neuron i is then computed as a weighted combination of the candidate activations:

$$\text{activation}_{l,i}(\mathbf{x}_{l,i}) = \sum_{j=1}^{|\mathcal{A}|} h_{l,i,j} \cdot \sigma_j(\mathbf{x}_{l,i}), \tag{5}$$

where $\mathbf{x}_{l,i}$ is the pre-activation input and $\sigma_j \in \mathcal{A}$ denotes the j-th activation function. The Gumbel–Max trick ensures that the perturbed $\arg\max$ produces unbiased samples from the categorical distribution

$$p_j = \frac{\exp(\alpha_{l,i,j})}{\sum_{k=1}^{|\mathcal{A}|} \exp(\alpha_{l,i,k})}, \qquad j = 1, \ldots, |\mathcal{A}|, \tag{6}$$

while the Gumbel–Softmax relaxation provides a differentiable approximation that enables gradient-based optimization of the logits $\boldsymbol{\alpha}_{l,i}$.

C. Phase 2: Mixed Network Construction

After training for a predetermined number of epochs in the selective phase, we extract a single activation per neuron by taking the maximum-logit choice:

$$\hat{\sigma}_{l,i} \;=\; \arg \max_{j \in \{1,\ldots,|\mathcal{A}|\}} \alpha_{l,i,j}, \tag{7}$$

where $\alpha_{l,i,j}$ is the learned logit for neuron i in layer l associated with activation function $\sigma_j \in \mathcal{A}$, and $\hat{\sigma}_{l,i} \in \mathcal{A}$ denotes the final chosen activation for neuron (l, i).

We then construct a fixed "mixed" network in which each neuron uses its selected activation. Let n_l denote the number of neurons in layer l. For computational efficiency, neurons with the same chosen activation are grouped together:

$$\mathcal{G}_{l,\sigma} \;=\; \{\, i \in \{1, \ldots, n_l\} \;:\; \hat{\sigma}_{l,i} = \sigma \,\}, \qquad \sigma \in \mathcal{A}. \tag{8}$$

In the forward pass of layer l, we first compute the pre-activation vector with a single affine transformation, using the trained weight matrix $\mathbf{W}_l \in \mathbb{R}^{n_l \times n_{l-1}}$ and bias vector $\mathbf{b}_l \in \mathbb{R}^{n_l}$ transferred from phase 1:

$$\mathbf{u}_l \;=\; \mathbf{W}_l \mathbf{x}_{l-1} + \mathbf{b}_l \;\in\; \mathbb{R}^{n_l}, \tag{9}$$

where $\mathbf{x}_{l-1} \in \mathbb{R}^{n_{l-1}}$ is the input from the previous layer.

The layer output is computed efficiently by processing each activation function group separately:

$$(\mathbf{y}_l)_i \;=\; \sigma\big((\mathbf{u}_l)_i\big), \qquad \forall i \in \mathcal{G}_{l,\sigma},\ \forall \sigma \in \mathcal{A}, \tag{10}$$

where for each activation function σ, we apply σ simultaneously to all neurons $i \in \mathcal{G}_{l,\sigma}$ that selected this activation

function. This vectorized implementation processes entire neuron groups at once: for each σ, extract the pre-activations of neurons in $\mathcal{G}_{l,\sigma}$, apply σ to this subvector, and assign the results back to their corresponding positions in $\mathbf{y}_l$.

After constructing the mixed network with fixed activation functions, we continue training the network for additional epochs. During this phase, the activation function selection remains fixed according to the choices made in Phase 1, while the network weights $\mathbf{W}_l$ and biases $\mathbf{b}_l$ continue to be optimized through standard backpropagation. This continued training allows the network parameters to adapt specifically to the selected activation configuration, often leading to improved performance as the weights can specialize for their assigned activation functions without the uncertainty introduced by the stochastic selection process of Phase 1.

IV. Experimental Setup

A. Data:

We conducted comprehensive experiments using the MNIST handwritten digit classification dataset, which consists of 70,000 grayscale images of digits 0-9, each with dimensions of 28×28 pixels. The dataset provides a well-established benchmark for evaluating neural network architectures while offering sufficient complexity to demonstrate the effectiveness of our adaptive activation approach.

To ensure robust evaluation and prevent overfitting, we employed stratified sampling to partition the original 60,000 training samples into two distinct subsets: 54,000 samples (90%) for training and 6,000 samples (10%) for validation. This stratified approach maintains the original class distribution across both subsets, ensuring balanced representation of each digit class. The official MNIST test set of 10,000 samples was used as the final test set for performance evaluation. All pixel values were normalized to the range [0, 1] by dividing by 255, and images were flattened to 784-dimensional vectors for input to fully connected networks.

B. Training SmartMixed

The first phase of SmartMixed training focuses on enabling each neuron to discover its optimal activation function through differentiable selection. For this analysis, we use a representative deep network with architecture [784, 768, 512, 512, 256, 256, 128, 10].

During this selective phase, we train the network for α epochs, where α represents the transition point between exploration and exploitation phases. In our experiments, we set $\alpha = 50$ epochs, though this parameter is adjustable based on the complexity of the architecture and convergence behavior. In future work, this parameter could be selected dynamically during training based on factors such as convergence in the loss function and the frequency of changes in activation function selections.

Figure 1 illustrates the training dynamics during the selective phase for the representative network architecture. The loss curves demonstrate healthy training behavior with a sharp initial decrease from approximately 2.0 to 1.5, followed by stabilization around epoch 30. The close alignment between training and validation curves throughout the process indicates effective learning without overfitting, confirming that the 50-epoch transition point provides sufficient time for activation preference stabilization.

The temperature parameter in the Gumbel-Softmax formulation plays a crucial role in controlling the sharpness of the learned activation selection. The hard discretization ensures that each neuron commits to a specific activation function for gradient computation based on its learned preferences. This design enables effective learning of activation preferences while maintaining gradient flow throughout the selective phase.

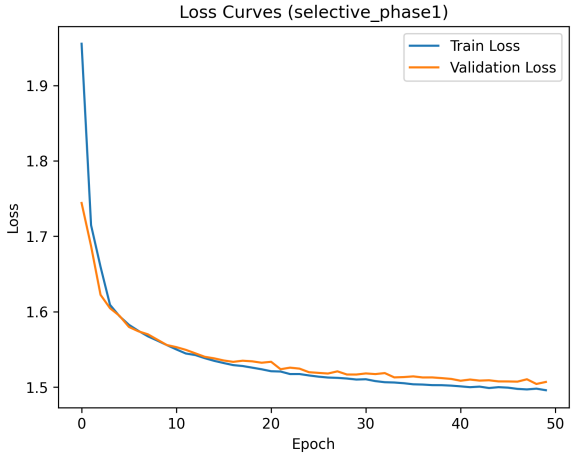

Fig. 1. Training and validation loss curves during Phase 1 (selective training).

Following Phase 1, we transition to Phase 2 by selecting the highest probability activation function for each neuron based on the learned logit preferences and reconstructing the network with these fixed activation choices. We then continue training this mixed network for an additional 350 epochs, primarily out of curiosity to observe the long-term training dynamics, though such an extended training period is not necessary for practical applications. Figure 2 demonstrates that the Phase 2 training exhibits healthy behavior with stable convergence and close alignment between training and validation curves, confirming the effectiveness of the mixed network architecture.

Table I presents the performance results across both phases of SmartMixed training.

TABLE I
Performance Comparison Between Phase 1 and Phase 2.

Training Phase	Validation Accuracy	Test Accuracy
Phase 1 (Selective)	95.67%	95.73%
Phase 2 (Mixed)	98.22%	98.03%

The results demonstrate the effectiveness of the two-phase approach, with Phase 2 achieving approximately 2-3% improvement in accuracy over Phase 1. This improvement is particularly significant because it occurs after fixing the activation functions, indicating that the learned activation choices provide

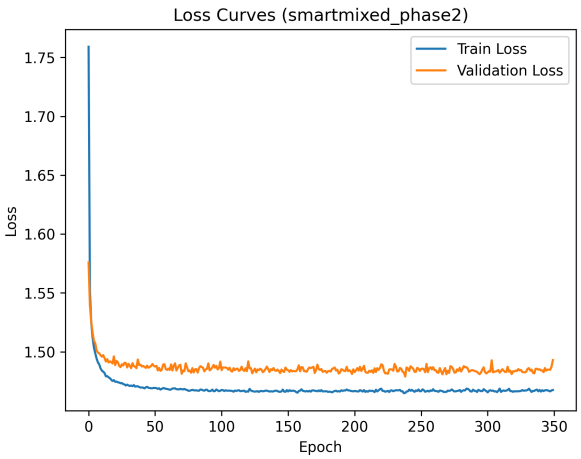

Fig. 2. Training and validation loss curves during Phase 2 (mixed network training) showing continued training for 350 epochs.

a better foundation for continued training. The stabilization of activation functions in Phase 2 is beneficial as any changes in activation function selection would require network weight readjustment due to different output ranges. By fixing the activation choices based on learned preferences, the network can focus entirely on optimizing the weights for the selected activation functions, leading to more stable and effective learning.

To better understand the dynamics of activation function selection during Phase 1, we explore how individual neurons choose their activation function in the following subsection.

C. Activation Function Selection Dynamics

For this analysis, we use the same representative network architecture from the previous section [784, 768, 512, 512, 256, 256, 128, 10], focusing exclusively on the six hidden layers and excluding the input and output layers from activation function selection. In the following analysis, "Layer 1" refers to the first hidden layer (768 neurons), "Layer 2" to the second hidden layer (512 neurons), and so forth, with "Layer 6" representing the final hidden layer (128 neurons) before the output layer. Figure 3 presents the final activation function distribution across all six hidden layers after 50 epochs of Phase 1 training, revealing distinct layer-wise preferences.

The results reveal clear patterns in activation function preferences across different network layers. Layer 1 neurons strongly prefer ReLU (37%) and Leaky ReLU (34%), accounting for over 70% of selections. Notably, Sigmoid is consistently avoided across all layers, likely due to its well-documented gradient vanishing problems that impede effective learning in deep networks. As network depth increases, preferences shift dramatically toward ELU and SELU, with Layer 6 showing the most pronounced preference for SELU (64%) and ELU (31%).

This analysis demonstrates that SmartMixed successfully enables neurons to discover functionally appropriate activation choices, with learned preferences reflecting the computational requirements of different network positions, early layers favor ReLU and Leaky ReLU while deeper layers prefer ELU and SELU. To validate the generalizability of these findings, we conducted extensive experiments across multiple feedforward neural network architectures of varying depths and widths. These experiments consistently revealed similar layer-wise activation preferences, confirming the robustness of our observations. To the best of our knowledge, such consistent layer-wise specialization patterns have not been explicitly reported in prior literature. The comprehensive performance analysis and architectural comparisons are presented in the following section.

D. Architecture-wise Activation Function Effectiveness

The objective of our architectural evaluation is not to find the best architecture that achieves the highest accuracy on the MNIST dataset, but rather to investigate two specific research questions: First, how does network topology, specifically the interplay between depth (number of layers) and width (neurons per layer), influence the activation function selection patterns learned by SmartMixed? Second, how does SmartMixed's adaptive approach compare against traditional fixed activation function strategies across diverse architectural configurations?

To address these questions systematically, we conducted experiments on 18 distinct architectures spanning a comprehensive range of network topologies: from shallow 3-layer networks to deep 14-layer architectures, with widths varying from 16 to 768 neurons per layer. For each architecture, we trained seven separate models, six using fixed activation functions (ReLU, Sigmoid, Tanh, Leaky ReLU, ELU, SELU) and one using our proposed SmartMixed approach, and ranked their performance based on test accuracy.

Table II presents the neural network architectures used in our experiments along with the top three approaches that achieved the highest accuracy on the test set.

Analysis of the activation function selections made by SmartMixed neurons across different layers and architectures reveals consistent patterns that align with our previous observations. Specifically, the distribution of selected activation functions confirms the layer-wise preferences identified in the previous subsection: deeper layers consistently favor ELU and SELU activation functions over alternatives such as ReLU, Sigmoid, and Tanh. This consistency across diverse architectural configurations demonstrates the robustness of the learned activation preferences and confirms the generalizability of these findings.

The results also reveal that SmartMixed demonstrates competitive performance by consistently appearing in the top 3 rankings across the majority of tested architectures. Notably, the performance differences between most activation functions are relatively small. However, Sigmoid consistently shows significantly lower performance, particularly in deeper architectures.

Figure 4 presents the Mean Reciprocal Rank (MRR) analysis, which provides a comprehensive measure of each acti-

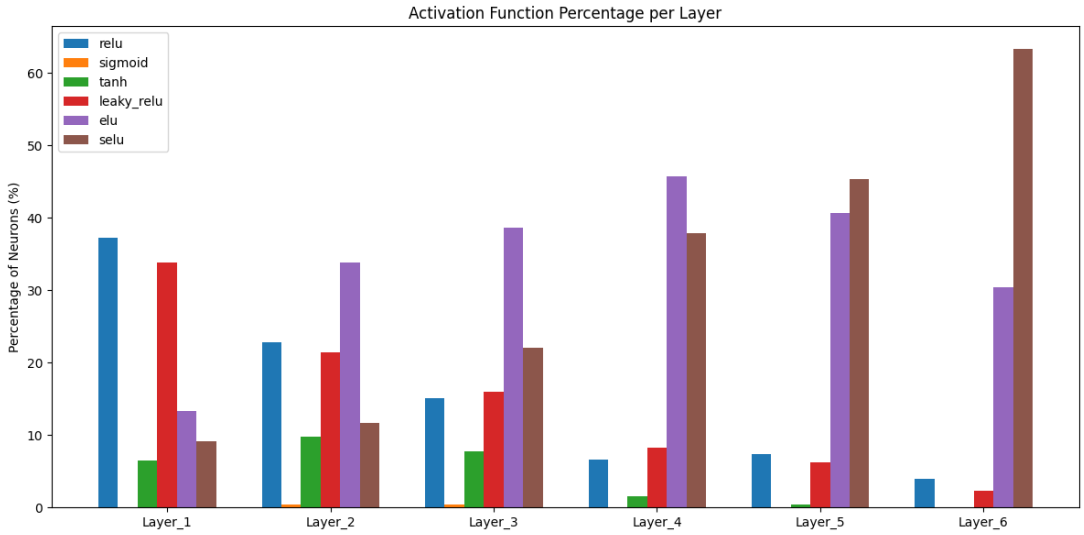

Fig. 3. Final activation function distribution across layers after Phase 1 training. Early layers favor ReLU and Leaky ReLU, while deeper layers increasingly prefer ELU and SELU activation functions.

TABLE II
ACTIVATION FUNCTION PERFORMANCE RANKINGS ACROSS 18 NETWORK ARCHITECTURES

Model Architecture	1st Rank	2nd Rank	3rd Rank
[784, 512, 10]	sigmoid	tanh	relu
[784, 512, 256, 128, 10]	smartmixed	leaky_relu	relu
[784, 512, 256, 128, 64, 10]	smartmixed	leaky_relu	tanh
[784, 512, 256, 256, 128, 10]	leaky_relu	elu	smartmixed
[784, 256, 128, 10]	smartmixed	tanh	relu
[784, 768, 10]	tanh	sigmoid	leaky_relu
[784, 768, 512, 10]	relu	leaky_relu	smartmixed
[784, 768, 512, 512, 10]	relu	smartmixed	leaky_relu
[784, 768, 512, 512, 256, 10]	leaky_relu	smartmixed	elu
[784, 768, 512, 512, 256, 256, 10]	leaky_relu	relu	smartmixed
[784, 768, 512, 512, 256, 256, 128, 10]	smartmixed	relu	selu
[784, 768, 512, 512, 256, 256, 128, 128, 10]	leaky_relu	smartmixed	relu
[784, 768, 512, 512, 256, 256, 128, 128, 64, 10]	relu	leaky_relu	selu
[784, 768, 512, 512, 256, 256, 128, 128, 64, 64, 10]	relu	leaky_relu	selu
[784, 768, 512, 512, 256, 256, 128, 128, 64, 64, 32, 10]	leaky_relu	relu	smartmixed
[784, 768, 512, 512, 256, 256, 128, 128, 64, 64, 32, 32, 10]	relu	smartmixed	leaky_relu
[784, 768, 512, 512, 256, 256, 128, 128, 64, 64, 32, 32, 16, 10]	leaky_relu	relu	selu
[784, 768, 512, 512, 256, 256, 128, 128, 64, 64, 32, 32, 16, 16, 10]	smartmixed	leaky_relu	relu

vation function's average performance across all architectures. MRR is calculated as:

$$\text{MRR} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{\text{rank}_i} \tag{11}$$

where n is the total number of architectures and rank_i is the rank of the activation function for architecture i. Higher values indicate better overall performance. The MRR analysis reveals the overall performance hierarchy across all architectures. Based on the ranking patterns, Leaky ReLU and ReLU show the strongest consistent performance, while SmartMixed demonstrates competitive results by consistently ranking in the top positions even when not achieving first place. The analysis shows that while no single activation function dominates universally, SmartMixed provides a robust adaptive solution that performs well across diverse architectural configurations.

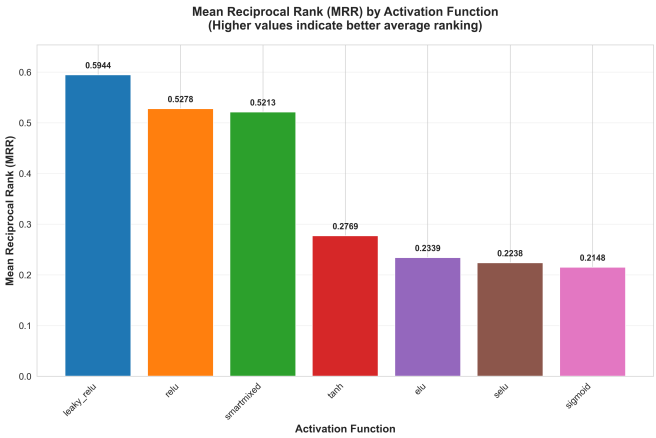

Fig. 4. Mean Reciprocal Rank analysis across all architectures. Leaky ReLU demonstrates the highest MRR, followed closely by ReLU and SmartMixed. SmartMixed shows consistent performance by ranking in top positions across diverse architectures.

Figure 5 complements the MRR analysis by showing the frequency distribution of rankings achieved by each activation function across all architectures. This visualization provides deeper insights into the consistency of performance for each method. The ranking distribution analysis reveals distinct performance patterns for each activation function across the 18 tested architectures. The results demonstrate a clear performance hierarchy: Leaky ReLU, ReLU, and SmartMixed emerge as the top-performing approaches, frequently competing for first-place rankings and consistently achieving positions within the top three. In contrast, traditional activation functions such as Sigmoid and Tanh consistently underperform, predominantly occupying lower-ranked positions across most architectures. This distribution pattern validates SmartMixed's effectiveness as an adaptive approach that can compete with well-established fixed activation functions.

To further understand the internal workings of SmartMixed networks, we analyzed the weight connectivity patterns between neurons with different activation functions across all architectures. Figure 6 presents a heatmap visualization of the average connection weights between neurons grouped by their activation function types, with Sigmoid excluded due to its rare selection frequency across the networks.

The heatmap reveals distinct patterns in inter-activation connectivity, showing how neurons with different activation functions connect through learned weight matrices. The source activation function (vertical axis) represents the sending neuron while the target activation function (horizontal axis) represents the receiving neuron, with color intensity indicating average connection weights. A notable observation is that neurons receiving inputs from ELU neurons exhibit consistently more positive average weights compared to other activation function pairs.

V. Conclusion

We introduced SmartMixed, a novel two-phase training strategy that enables practical per-neuron activation function learning in neural networks. Our approach successfully bridges the gap between adaptive and fixed activation approaches, providing the benefits of both while mitigating their respective limitations. The process encompasses two phases: the selective phase enables thorough exploration of activation function choices using differentiable Gumbel-Softmax sampling, while the mixed phase exploits learned preferences through efficient vectorized operations. Experimental results on MNIST demonstrate the effectiveness of our approach, with SmartMixed ranking among the top three performing approaches across 18 diverse network architectures, validating its robustness and adaptability.

Most significantly, our analysis reveals intrinsic activation function preferences that emerge at different network depths. We discovered that neurons in different layers exhibit distinct and consistent preferences: early layers strongly favor ReLU and Leaky ReLU activation functions while deeper layers progressively shift toward ELU and SELU functions. This layer-wise specialization pattern remained consistent across diverse architectural configurations. These findings provide novel insights into the functional diversity within neural architectures and demonstrate that individual neurons can intelligently adapt their activation functions based on their positional role in the network.

SmartMixed opens new avenues for neural architecture optimization by demonstrating that sophisticated architectural choices can be learned during training without sacrificing deployment efficiency. The principles underlying our approach are broadly applicable and could inspire similar staged optimization strategies for other architectural components.

While our current work demonstrates the effectiveness of SmartMixed on the MNIST dataset using feedforward neural networks, several promising directions emerge for future research. First, evaluating SmartMixed on more complex and diverse datasets and domain-specific datasets would provide insights into the generalizability of our observed activation function preferences across different data modalities and complexity levels. Additionally, extending our approach to different types of neural networks would assess whether what we discovered could hold across various architectural paradigms.

References

[1] A. Omidvar, H. Jiang, and A. An, "Using neural network for identifying clickbaits in online news media," in *Annual International Symposium on Information Management and Big Data*. Springer, 2018, pp. 220–232.

[2] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.

[3] A. L. Maas, A. Y. Hannun, A. Y. Ng *et al.*, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, no. 1. Atlanta, GA, 2013, p. 3.

[4] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, vol. 4, no. 5, p. 11, 2015.

[5] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," *Advances in neural information processing systems*, vol. 30, 2017.

[6] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," *arXiv preprint arXiv:1710.05941*, 2017.

[7] A. Apicella, F. Donnarumma, F. Isgrò, and R. Prevete, "A survey on modern trainable activation functions," *Neural Networks*, vol. 138, pp. 14–32, 2021.

[8] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," *Neural networks*, vol. 107, pp. 3–11, 2018.

[9] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia, "Incorporating second-order functional knowledge for better option pricing," *Advances in neural information processing systems*, vol. 13, 2000.

[10] F. Manessi, A. Rozza, and M. Manzo, "Dynamic graph convolutional networks," *Pattern Recognition*, vol. 97, p. 107000, 2020.

[11] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.

[12] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[13] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144*, 2016.

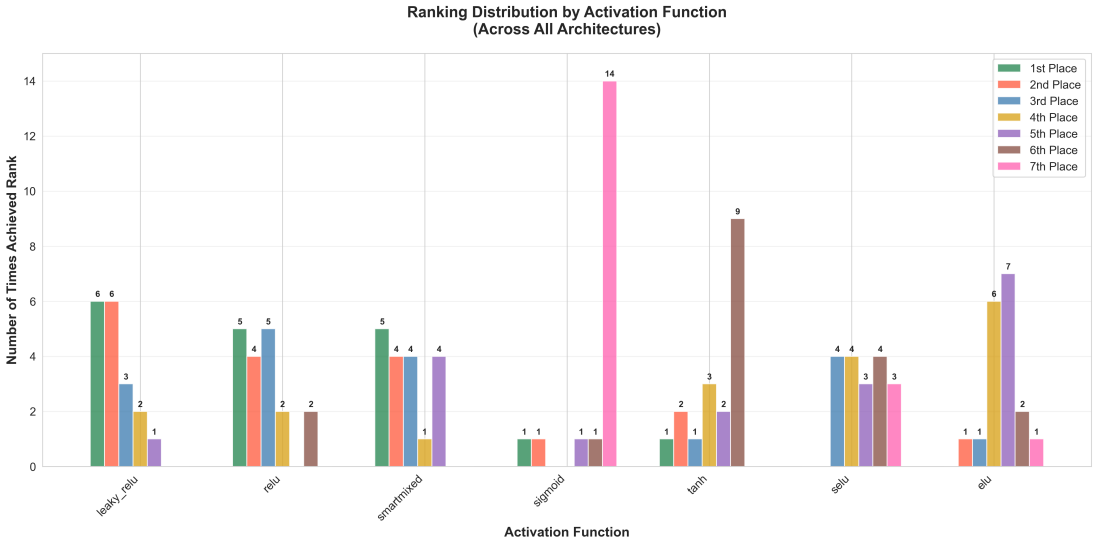

Fig. 5. Ranking distribution showing the frequency of each rank position (1st through 7th) achieved by each activation function across all 18 architectures. SmartMixed demonstrates consistent performance with frequent appearances in top positions.

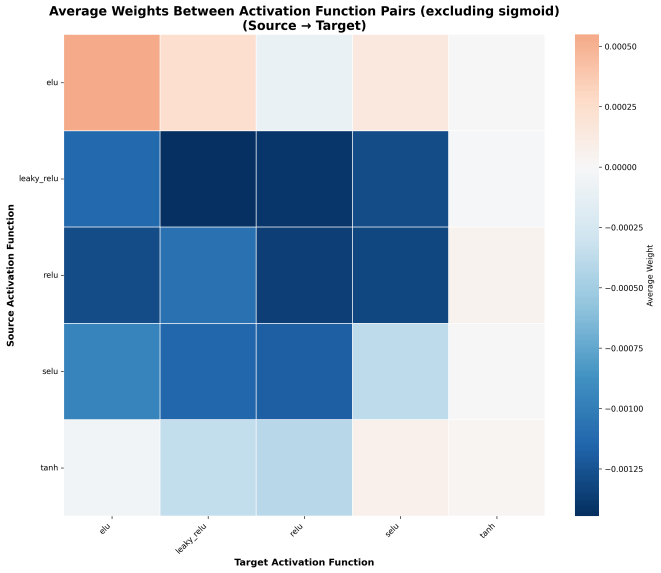

Fig. 6. Heatmap of average connection weights between neurons grouped by activation function types across all 18 architectures. Sigmoid is excluded due to infrequent selection. The visualization shows weight patterns between source neurons (vertical axis) and target neurons (horizontal axis), revealing connectivity preferences between different activation function types.