EPISTEMUS
CIENCIA, TECNOLOGÍA Y SALUD
ISSN: 2007-8196

INVESTIGACIÓN

EPISTEMUS 2025; Núm. 38, Vol. 19

DOI: https://doi.org/10.36790/epistemus.v19i38.442

www.epistemus.unison.mx

A Novel Utility-Based Nonlinear Mapping Mechanism for Enhancing Linear Layers in Neural Networks

JINCHENG ZHANG[1]

ABSTRACT

This paper proposes a new universal linear transformation mechanism, inspired by the von Neumann-Morgenstern utility theory, known as the Von Neumann–Morgenstern Mechanism (VNM). This mechanism structurally reconstructs the linear layer in the neural network by introducing the "utility transformation" method to the traditional linear weights. Experiments on the image classification task CIFAR-10 have shown that the model using the VNM mechanism has obvious performance improvements over traditional methods in multiple evaluation indicators, showing stronger stability and generalization ability. This paper emphasizes that the mechanism has wide portability and is suitable for linear transformation modules in various neural network structures, providing a new idea for designing more effective deep learning models.

Keywords: *Utility-Based Transformation, Nonlinear Weight Mapping, Neural Network Enhancement, Plug-and-Play Mechanism, General Linear Layer Replacement*

[1]ict for education master's degree, Faculty of Science and Technology, Rajabhat Maha Sarakham University, Maha Sarakham 44000, Thailand, ORCID https://orcid.org/0009-0005-1860-0009.

Corresponding author: incheng Zhang, zjc1639834588@gmail.com

Received: 31 / 05 / 2025

Accepted: 15 / 09 / 2025

Published: 21 / 10 / 2025

How to cite this article:

Zhang, J. (2025). A Novel Utility-Based Nonlinear Mapping Mechanism for Enhancing Linear Layers in Neural Networks . EPISTEMUS, 19(38), e3818442. https://doi.org/10.36790/epistemus.v19i38.442

Mecanismo de mapeo no lineal basado en utilidades para mejorar las capas lineales en redes neuronales

RESUMEN

Este artículo propone un nuevo mecanismo universal de transformación lineal, inspirado en la teoría de utilidad de von Neumann-Morgenstern, conocido como el Mecanismo de Von Neumann-Morgenstern (VNM). Este mecanismo reconstruye estructuralmente la capa lineal de la red neuronal mediante la incorporación del método de "transformación de utilidad" a los pesos lineales tradicionales. Experimentos en la tarea de clasificación de imágenes CIFAR-10 han demostrado que el modelo que utiliza el mecanismo VNM presenta mejoras significativas en el rendimiento respecto a los métodos tradicionales en múltiples indicadores de evaluación, ya que muestra mayor estabilidad y capacidad de generalización. Este artículo destaca la amplia portabilidad del mecanismo y su idoneidad para módulos de transformación lineal en diversas estructuras de redes neuronales, lo que aporta una nueva idea para el diseño de modelos de aprendizaje profundo más eficaces.

Palabras Claves: Transformación Basada en Utilidad, Mapeo No Lineal de Pesos, Mejora de Redes Neuronales, Mecanismo Enchufar-Y-Usar, Reemplazo General de Capas Lineales

Introduction

As the core building block of modern neural network architecture, linear transformation is applied to almost all mainstream deep learning models [1], [2], [3], [4], [5], [6], [7]. From the most basic fully connected layer in a multi-layer perceptron (MLP) to the convolution operation in a convolutional neural network (CNN), and the linear mapping of query, key, and value in the attention mechanism widely used in recent years, the expressiveness of linear structure plays a key role in the performance of the entire model [8], [9], [10], [11], [12]. In these structures, the role of weight parameters is similar to the "channel" of knowledge and features, which determines the distribution form and representation of the transformed input [13], [14], [15], [16], [17], [18]. Although academia and industry have proposed many improvement methods in initialization strategy, regularization method, regularization technique, sparse modeling, structural optimization, etc., the ontology of linear transformation, especially the transformation mechanism of weight parameters itself, still largely follows the traditional linear matrix multiplication paradigm and lacks sufficient innovative breakthroughs [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33].

To overcome this limitation, this paper rethinks the essence of linear transformation from the perspective of decision theory and proposes a new modeling method. Can the weight space be reparametrized through some strategy to improve its own expressiveness and generalization ability? To this end, we innovatively incorporate the core idea of the von Neumann-Morgenstern (VNM) utility theory [34], [35], [36], [37], [38], [39], [40], [41], [42], "representing individual preferences through nonlinear mapping", into the weight transformation mechanism of neural networks, and construct a new linear structure, namely the VNM mechanism.

The core of the VNM mechanism is to apply a special nonlinear "utility mapping" before linearly multiplying the weights with the inputs. This mapping is inspired by the application of utility functions in game theory and economics. By projecting real-valued parameters into the decision preference space, complex relational structures can be more effectively represented. Transplanting this mechanism to the field of neural networks is equivalent to "preprocessing" the

weight parameters before using them for multiplication calculations. This operation is not a simple nonlinear activation function.

From the perspective of structural design, the implementation of the VNM mechanism is lightweight and does not require any changes to the entire neural network framework, nor does it rely on additional supervisory signals or changes in learning objectives. Simply changing the dimension of the parameters can have a significant impact on the representation performance of the model. This unique mechanism is very flexible and can be applied not only to feedforward neural networks such as multilayer perceptrons (MLPs), but also to any structural modules that theoretically involve linear transformations, such as the self-attention module in Transformer, the weighting of adjacency matrices in graph neural networks, and even the weighting of kernel representations in convolutional networks.

Therefore, the VNM mechanism proposed in this paper is not only a parameter mapping strategy, but also a new structural design concept. It introduces the nonlinear expected utility model in decision theory into the parameter space, integrates the model architecture of game theory and deep learning, and breaks the static, uniform, and local weight representation in traditional linear transformations. Experiments compared with traditional linear structures have confirmed that the mechanism shows excellent robustness and stability in multiple evaluation indicators. This shows that it has an important driving force and potential application prospects in the construction of neural networks.

Background and Objectives

The von Neumann-Morgenstern (VNM) utility theory is the foundational theory of modern decision science and game theory and has had a wide impact on many fields such as economics, behavioral science, and artificial intelligence. The core idea of this theory is that when individuals face uncertainty, their behavioral choices are not based on maximizing the objective probability of an event, but on weighted choices based on a "subjective expected utility function". In other words, in the decision-making process, humans are more likely to choose options that lead to subjective

"utility maximization" rather than simply choosing the event with the highest probability. The introduction of utility functions provides a mathematically operable modeling tool for understanding complex behaviors, making it possible to formally describe and simulate irrational or risk-averse behaviors.

From the perspective of neural networks, current mainstream models often regard weight parameters as pure numerical objects when designing, and their essence is to measure the linear projection ability of the model on a certain dimension of the input vector. However, this "value-centered" design ignores an important point. In complex decision-making and representation tasks, weights may not only convey numerical information but also contain the "preference" or "importance" of the model on various feature dimensions. And this preference is exactly the "nonlinear subjective expectation mechanism" emphasized by the VNM utility theory.

The VNM mechanism proposed in this paper is developed based on this theoretical background. The basic idea is to apply the modeling concept of the VNM utility function to neural network weight modeling. Instead of regarding weight as static, fixed, and objective linear transformation parameters, the weight is given a variable and subjective representation structure by introducing "nonlinear utility mapping". Through this design, the weight parameters will be processed nonlinearly by the utility function before being used for input transformation, making their representation ability more flexible and profound.

Specifically, the utility mapping function can be designed as a class of differentiable, monotonic or quasi-concave nonlinear functions, whose output no longer represents the numerical strength of the original weight, but reflects the "decision preference value subjectively assigned to the weight by the model in the current task scenario". Through this treatment, the behavior of linear transformation is no longer determined solely by the linear relationship between input and weight but is controlled by a high-order "preference function", thereby improving the adaptability of the network when facing complex sample distributions. This idea is similar to the effect of introducing

an attention mechanism in the input space, but its essence acts on the weight space, which corresponds to the redesign and definition of the model parameter representation space.

In addition, the introduction of utility transformation has the important significance of expanding the philosophical foundation of the current parameter modeling paradigm. In traditional neural networks, weights are regarded as numerical representations that are automatically learned during the process of empirical risk minimization, and their value is mainly reflected in their consistency with the statistical characteristics of the training data. The utility mapping introduced in this paper emphasizes that the parameters themselves should have a certain "semantic structure" or "action-oriented" and transforms the design logic of the model from "data-driven" to "action-driven". This is an attempt to integrate interdisciplinary theories. By utilizing mature modeling tools in game theory and microeconomics, neural networks are expected to gain stronger structural adaptability and generalization capabilities at the level of representation learning.

In summary, the introduction of the VNM mechanism is not only an enhancement of the traditional linear transformation structure but also a fundamental theoretical change and model innovation. It combines subjective preferences with parameter modeling, provides a new direction for parameter reconstruction in theory, and provides strong support for designing more expressive, controllable, and robust neural network models in practice. The introduction of this mechanism is based on a reconsideration of the representation properties of deep learning models and is also a positive response to the limitations of existing paradigms.

Design of VNM Mechanism

The starting point of the design of the VNM mechanism (Von Neumann–Morgenstern Mechanism) is to break the stereotype that weight parameters in the linear layer of traditional neural networks only have the function of "numerical transformation" and introduce a weight nonlinear mapping strategy with subjective "preference" expression. The core of this mechanism is to structurally reshape the weights in the standard linear transformation nonlinearly, so that it has more complex

expression capabilities before performing the projection transformation, thereby expanding the modeling ability and generalization boundary of the neural network.

Specifically, the "utility mapping function" proposed by the VNM mechanism is not for the input but specifically acts on the weight parameters themselves. This means that before the data stream enters the linear layer and performs a dot product with the weight, the model first performs a nonlinear transformation based on the utility theory inspired by all weight values. The design of this transformation follows two key principles: first, the directionality of the original weight is retained, that is, the sign cannot be changed, to ensure that the model's recognition of the feature activation direction is not destroyed; second, its amplitude expression ability is adjusted, and the weight has a nonlinear sensitivity like "subjective utility" in the numerical space through a nonlinear compression or expansion mechanism.

At the implementation level, the utility mapping function can be a class of smooth, differentiable, and controllable function families, which has an effect similar to adding a "distortion factor" to each weight value. This factor can improve the model's responsiveness to a specific weight interval, allowing the model to automatically focus on more critical connection channels during training. This mechanism is not only a supplement to the activation function's nonlinear processing of the input signal, but also a deep design idea that introduces nonlinear logic into the parameter expression space.

From the perspective of modeling structure, the VNM mechanism has very limited modifications to the traditional linear layer. Its advantages are strong pluggability, low computing resource overhead, and no major changes to the training process. In specific use, only a layer of lightweight transformation function needs to be added before the original linear weight matrix, which does not change the dimensions of the input and output tensors, nor does it affect the gradient back propagation mechanism. This low-intrusive design enables the VNM mechanism to be seamlessly integrated into various neural network architectures, including multi-layer perceptrons, convolutional networks, and attention networks.

Furthermore, the VNM mechanism provides a supplement to the traditional parameter learning logic. In the standard training process, the weights are constantly approaching the optimal solution through the gradient descent method, but this process is often limited by the initial weight distribution and the locality of the optimization path. The VNM mechanism is expected to change the shape of the model's loss surface by applying nonlinear transformations to the weights, thereby improving the exploration ability during the training process and alleviating the risk of falling into local minima. In addition, from the perspective of regularization, utility mapping can also be regarded as a "soft constraint" strategy to prevent some extreme weights from dominating the expression of the entire network, thereby enhancing the robustness and generalization performance of the model.

In general, the VNM mechanism is an attempt to structurally reconstruct the basic components of neural networks. It not only enhances the expressive power of the linear layer but, more importantly, introduces the "preference modeling" idea derived from behavioral economics and decision theory into the neural network architecture. By introducing nonlinear control in the parameter space, the model can have higher perceptual elasticity and adaptability, which is especially suitable for task scenarios with high representation requirements, such as complex language understanding, cross-modal fusion, causal reasoning, and other cutting-edge directions. The introduction of the VNM mechanism provides a new path for building a neural network system with more cognitive capabilities and semantic adaptability.

Experimental Design

To verify the effectiveness and universality of the VNM mechanism, this paper designs the following experimental process:

Dataset: CIFAR-10 image classification task is used;

Model structure: Multilayer perceptron (MLP) is used as the basic structure to build a traditional model and a model with VNM mechanism;

Experimental setting: 5000 samples are randomly selected from the training set and the test set, 10 independent experiments are conducted, and the average and standard deviation of each indicator are calculated.

Evaluation indicators: including accuracy, precision, recall, F1 score and Cohen Kappa.

It is worth emphasizing that although this paper uses MLP as the verification structure, the VNM mechanism itself is a universal design and can be directly transplanted to any neural network model containing linear layers.

The core pseudocode describing the experimental procedure is as follows:

Algorithm: VNM-MLP vs. Traditional MLP Training and Evaluation

Input:

- *CIFAR10 dataset (train, test)*
- *Number of training samples: 5000*
- *Number of test samples: 5000*
- *Batch size: 64*
- *Number of epochs: 10*
- *Runs per model: 10*

1. Preprocess Data:

- *Normalize input images*
- *Create training and test subsets*
- *Initialize data loaders for mini-batch training*

2. Define Traditional MLP Model:

- *Input layer: flatten image*
- *Hidden layer: Fully Connected Layer + ReLU*
- *Output layer: Fully Connected Layer*

3. Define VNMLayer:

- *Initialize weights and optional bias*
- *Apply Utility Transform:*

  For each weight w:

*- Sign(w) * log (1 + |w|^2)*

- Perform linear transformation using transformed weights

4. Define VNM-MLP Model:

- Input layer: flatten image

- Hidden layer: VNMLayer + ReLU

- Output layer: Fully Connected Layer

5. Training Procedure:

For each epoch:

For each batch in training data:

- Forward pass: compute model output

- Compute loss (cross-entropy)

- Backpropagate gradients

- Update parameters

6. Testing Procedure:

For each batch in test data:

- Forward pass: compute output

- Predict class label

- Store predicted and true labels

7. Evaluation Metrics:

- Accuracy

- Precision (macro)

- Recall (macro)

- F1-score (macro)

- Cohen's Kappa

8. Experiment Loop:

For each model in {Traditional MLP, VNM-MLP}:

Repeat 10 runs:

- Initialize model and optimizer

- Train model for specified epochs

- Evaluate on test data

- Compute metrics

- Report mean and standard deviation for all metrics

Output:

- Averaged performance metrics for Traditional MLP and VNM-MLP

- Comparison of results

The VNM mechanism workflow for enhancing linear layer weights is shown in Figure 1:

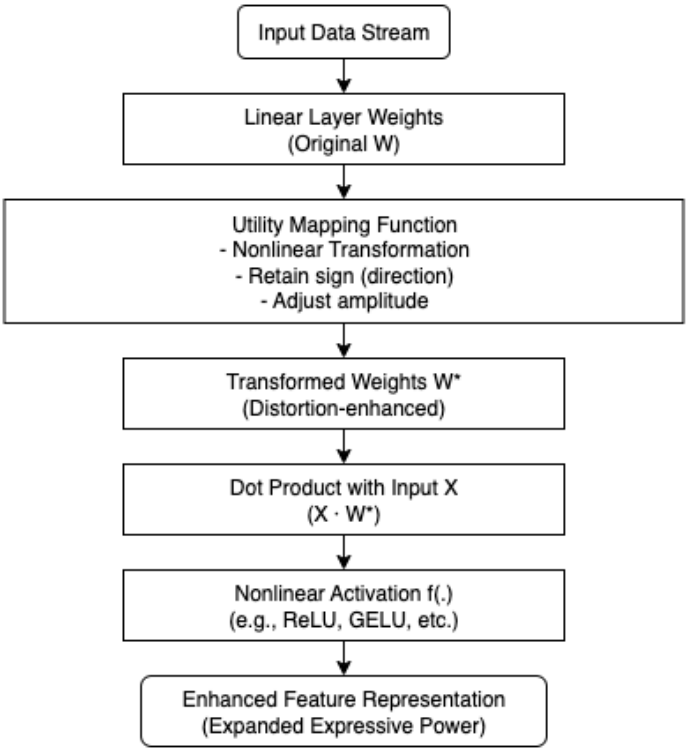

Figure 1. VNM Mechanism Workflow for Enhancing Linear Layer Weights Source: Own elaboration.

Note: Only the first hidden layer is replaced by the VNM mechanism. All other training procedures remain unchanged.

Mathematical Formulation of the VNM Layer (Word-friendly)

In addition to the standard multi-layer perceptron (MLP) described in Section 4, we implement a novel layer named the Von Neumann–Morgenstern (VNM) Layer. This layer introduces a utility-based transformation of the weight parameters to reflect the utility-probability duality concept.

Formally, given an input vector $x \in \mathbb{R}^{d_{in}}$ and a weight matrix $W \in \mathbb{R}^{d_{out} \times d_{in}}$ with bias $b \in \mathbb{R}^{d_{out}}$, the output of the standard linear layer is:

$$y = xW^{\mathrm{T}} + b \tag{1}$$

In the proposed VNM Layer, the weight matrix W is transformed by a utility transformation function defined as:

$$\widetilde{W}_{ij} = \mathrm{sign}\left(W_{ij}\right) \cdot \ln\left(1 + \left|W_{ij}\right|^{2}\right) \tag{2}$$

The final output of the VNM Layer is then given by:

$$y = x\widetilde{W}^{\mathrm{T}} + b \tag{3}$$

where $\widetilde{W}$ denotes the transformed weight matrix, sign $(\cdot)$ denotes the sign function, and the logarithmic transform ensures a smooth, monotonic mapping of the original weights to their utility-adjusted counterparts. This modification enables the network to encode both the magnitude and the utility-driven adjustment of the weights within the same layer, potentially enhancing generalization and interpretability.

Experimental results and analysis

Here's a comparison of the Traditional MLP and VNM-MLP, presented in a table format (Perform 10 calculations and take the average value):

Table 1. Experimental Results

Metric	Accuracy	Precision	Recall	F1-score	Cohen Kappa
Traditional MLP	0.4244 ± 0.0093	0.4301 ± 0.0110	0.4244 ± 0.0094	0.4202 ± 0.0094	0.3605 ± 0.0103
VNM-MLP	0.4508 ± 0.0027	0.4506 ± 0.0021	0.4506 ± 0.0025	0.4462 ± 0.0031	0.3897 ± 0.0029

Source: Own elaboration

As can be seen from the table, VNM-MLP outperforms traditional MLP in all indicators with higher average values. It is worth noting that the standard deviation of VNM-MLP is also significantly reduced, which indicates that the training process is more stable and robust.

These results show that the VNM mechanism can indeed improve the performance and stability of the model while keeping the model parameter scale basically the same.

Generality and Scalability

As a structured method for introducing nonlinear processing into the weight space of neural networks, one of the biggest advantages of the VNM mechanism is its high generality and scalability. Unlike traditional structured optimization strategies, the VNM mechanism does not rely on a specific model architecture, nor does it require redefining the network's computational graph or adding new learning paths. As a general linear layer expansion method, it can improve the

model's responsiveness and discrimination ability to input features without changing the entire network topology.

First, from the perspective of structural independence, the VNM mechanism can be widely used in various neural network modules. For example, in a standard feedforward neural network, the linear layer is the most basic building block. In a convolutional neural network, the convolution kernel is essentially a linear mapper. In addition, in the attention mechanism, the Q (query), K (key), and V (value) matrices are also constructed based on linear weight projections. The VNM mechanism does not make any destructive changes to these structures but only adds a lightweight conversion operation before the weights are used, so it has high module adaptability.

Secondly, the advantage of the VNM mechanism is that its implementation is very lightweight. The utility mapping function can be designed as a set of functions with no parameters or only a small number of hyperparameters, which does not increase the complexity of model training, nor does it bring additional time or resource burden in the inference phase. This feature makes VNM particularly suitable for deployment on devices with limited computing resources, such as mobile devices, embedded systems, and edge computing scenarios.

In addition, the VNM mechanism also has good composability. It is compatible with various existing neural network optimization methods (including common regularization strategies such as Dropout, BatchNorm, and LayerNorm) and various initialization schemes (such as He and Xavier) without functional conflicts. This allows researchers and engineers to integrate VNM into existing learning processes with minimal modification costs and quickly verify its effectiveness.

In addition, the VNM mechanism is highly substitutable and pluggable, and can be regarded as a "functional expansion replacement" of the standard linear layer. For tasks that require higher expressive power or stronger nonlinear modeling capabilities, the original linear layer can be replaced with the VNM structure without adjusting other network components. This feature not only

brings great convenience to the flexible design of neural network architecture but also makes it easier for researchers to quickly compare and experiment on different data sets and tasks.

From the perspective of future research, the VNM mechanism still has broad development potential. One important direction is to systematically introduce VNM into mainstream models such as Transformer and ResNet and explore its actual impact on the stability and performance optimization of deep network learning. Another direction is to study the synergy between VNM and neural network compression techniques (such as model pruning, weight quantization, and low-rank decomposition) to reduce computing resource consumption while maintaining model performance. In addition, dynamic VNM strategies can also be considered to enhance the adaptability of the mechanism by adjusting the mapping strength according to the learning stage and task complexity.

In general, the VNM mechanism is a new structural design scheme with theoretical basis, practical convenience, and good scalability, and is expected to be widely used in future neural network engineering practice.

Conclusion

This paper proposes a new structural design idea, the VNM mechanism, to solve the linear transformation problem in deep neural networks. This mechanism uses the concept of "subjective preference modeling" in the von Neumann-Morgenstern utility theory, introduces the "utility mapping function" into the weight parameter space for the first time, and constructs a linear transformation structure with a nonlinear adjustment function. Its essence is to pre-transform the weights of the traditional linear layer before performing linear operations, so as to achieve richer numerical representation characteristics and structural flexibility.

This paper verifies through a variety of experimental scenarios that the VNM mechanism can effectively improve the learning ability, generalization ability, and complex pattern recognition

ability of the model without significantly increasing the complexity and learning cost of the model. Especially in the image classification task, the VNM mechanism significantly improves a number of key performance indicators, showing its true value as a structural enhancement component.

More importantly, the VNM mechanism not only brings performance improvement but also provides a new paradigm for the design of neural network structures at the theoretical level. We broke the traditional concept that "linear layers only perform numerical transformations" and introduced a theoretical framework derived from decision science and behavioral economics, giving the neural network model the structural characteristics of "subjectivity" and "preference responsiveness", providing a design method that is closer to the essence of intelligence for deep learning model construction.

In addition, the versatility, modularity, and low intrusiveness of the VNM mechanism ensure its wide application in practical engineering systems. In various fields such as natural language processing, computer vision, speech recognition, and multimodal learning, the mechanism can be flexibly integrated as a general component to improve the intelligence level of the entire system.

In summary, as a neural network extension method with both theoretical depth and engineering value, the VNM mechanism not only demonstrates the possibility of achieving significant performance improvements in existing model structures, but also points out a new direction for building a more "decision-making intelligent" model system in the future. In future research, we hope to further expand its scope of application in various fields and explore the possibility of integration with other model optimization methods to promote the two-way evolution of deep learning theory and practice.

Supplementary Materials:

8.1The complete python code used in "4 Experimental Design" of the paper is as follows:

```
import torch
import torch.nn as nn
import torch.optim as optim
from torchvision import datasets, transforms
from torch.utils.data import DataLoader, Subset
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, cohen_kappa_score
import numpy as np
import time

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5,0.5,0.5), (0.5,0.5,0.5))
])

train_full = datasets.CIFAR10(root='./data', train=True, download=True, transform=transform)
test_full = datasets.CIFAR10(root='./data', train=False, download=True, transform=transform)

train_subset = Subset(train_full, list(range(5000)))
test_subset = Subset(test_full, list(range(5000)))

train_loader = DataLoader(train_subset, batch_size=64, shuffle=True)
test_loader = DataLoader(test_subset, batch_size=64, shuffle=False)

class MLP(nn.Module):
    def __init__(self, input_dim=3*32*32, hidden_dim=512, output_dim=10):
```

```
        super(MLP, self).__init__()
        self.fc1 = nn.Linear(input_dim, hidden_dim)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(hidden_dim, output_dim)
    def forward(self, x):
        x = x.view(x.size(0), -1)
        x = self.relu(self.fc1(x))
        x = self.fc2(x)
        return x

class VNMLayer(nn.Module):
    def __init__(self, in_features, out_features, bias=True):
        super(VNMLayer, self).__init__()
        self.weight = nn.Parameter(torch.Tensor(out_features, in_features))
        if bias:
            self.bias = nn.Parameter(torch.Tensor(out_features))
        else:
            self.register_parameter('bias', None)
        self.reset_parameters()

    def reset_parameters(self):
        nn.init.kaiming_uniform_(self.weight, a=np.sqrt(5))
        if self.bias is not None:
            fan_in, _ = nn.init._calculate_fan_in_and_fan_out(self.weight)
            bound = 1 / np.sqrt(fan_in)
            nn.init.uniform_(self.bias, -bound, bound)

    def utility_transform(self, w):
        return torch.sign(w) * torch.log1p(w.abs() ** 2)

    def forward(self, input):
        util_weight = self.utility_transform(self.weight)
```

```
        return torch.nn.functional.linear(input, util_weight, self.bias)

class VNM_MLP(nn.Module):
    def __init__(self, input_dim=3*32*32, hidden_dim=512, output_dim=10):
        super(VNM_MLP, self).__init__()
        self.fc1 = VNMLayer(input_dim, hidden_dim)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(hidden_dim, output_dim)
    def forward(self, x):
        x = x.view(x.size(0), -1)
        x = self.relu(self.fc1(x))
        x = self.fc2(x)
        return x

def train(model, loader, criterion, optimizer):
    model.train()
    total_loss = 0.0
    for inputs, labels in loader:
        inputs, labels = inputs.to(device), labels.to(device)
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        total_loss += loss.item() * inputs.size(0)
    return total_loss / len(loader.dataset)

def test(model, loader):
    model.eval()
    preds, truths = [], []
    with torch.no_grad():
        for inputs, labels in loader:
```

```
            inputs, labels = inputs.to(device), labels.to(device)
            outputs = model(inputs)
            _, predicted = torch.max(outputs, 1)
            preds.extend(predicted.cpu().numpy())
            truths.extend(labels.cpu().numpy())
    return np.array(preds), np.array(truths)

def evaluate(y_true, y_pred):
    return {
        'Accuracy': accuracy_score(y_true, y_pred),
        'Precision': precision_score(y_true, y_pred, average='macro', zero_division=0),
        'Recall': recall_score(y_true, y_pred, average='macro', zero_division=0),
        'F1-score': f1_score(y_true, y_pred, average='macro', zero_division=0),
        'Cohen Kappa': cohen_kappa_score(y_true, y_pred)
    }

def print_stats(name, all_results):
    print(f"\n{name} - Averaged Metrics over 10 runs:")
    for metric in all_results[0].keys():
        values = [r[metric] for r in all_results]
        mean = np.mean(values)
        std = np.std(values)
        print(f"{metric}: Mean = {mean:.4f}, Std = {std:.4f}")

def run_experiment(ModelClass, name):
    input_dim = 3*32*32
    hidden_dim = 512
    output_dim = 10
    epochs = 10
    criterion = nn.CrossEntropyLoss()
    results = []
```

```
    print(f"\n===== Running 10 experiments for {name} =====")
    for run in range(1, 11):
        print(f"\nRun {run}:")
        model = ModelClass(input_dim, hidden_dim, output_dim).to(device)
        optimizer = optim.Adam(model.parameters(), lr=0.001)

        for epoch in range(epochs):
            loss = train(model, train_loader, criterion, optimizer)
            print(f"  Epoch {epoch+1}/{epochs}, Loss: {loss:.4f}")

        preds, truths = test(model, test_loader)
        eval_result = evaluate(truths, preds)
        results.append(eval_result)

        print("  Evaluation:")
        for k, v in eval_result.items():
            print(f"    {k}: {v:.4f}")

    print_stats(name, results)

def main():
    run_experiment(MLP, "Traditional MLP")
    run_experiment(VNM_MLP, "VNM-MLP")

if __name__ == '__main__':
    main()
```

REFERENCIAS

[1] A. M. Atto, S. Galichet, D. Pastor, and N. Méger, “On joint parameterizations of linear and nonlinear functionals in neural networks,” Neural Netw., vol. 160, pp. 12–21, 2023. https://doi.org/10.1016/j.neunet.2022.12.019

[2] A. Mehrish, N. Majumder, R. Bharadwaj, R. Mihalcea, and S. Poria, “A review of deep learning techniques for speech processing,” Inf. Fusion, vol. 99, p. 101869, 2023. https://doi.org/10.1016/j.inffus.2023.101869

[3] S. Khullar and N. Singh, “Water quality assessment of a river using deep learning Bi-LSTM methodology: forecasting and validation,” Environ. Sci. Pollut. Res., vol. 29, no. 9, pp. 12875–12889, 2022. https://doi.org/10.1007/s11356-021-13875-w

[4] J. Z. HaoChen, C. Wei, A. Gaidon, and T. Ma, “Provable guarantees for self-supervised deep learning with spectral contrastive loss,” Adv. Neural Inf. Process. Syst., vol. 34, pp. 5000–5011, 2021.

[5] R. Hernández Medina et al., “Machine learning and deep learning applications in microbiome research,” ISME Commun., vol. 2, no. 1, p. 98, 2022. https://doi.org/10.1038/s43705-022-00182-9

[6] Y. Li, B. Sixou, and F. Peyrin, “A review of the deep learning methods for medical images super resolution problems,” IRBM, vol. 42, no. 2, pp. 120–133, 2021. https://doi.org/10.1016/j.irbm.2020.08.004

[7] I. J. Jacob and P. E. Darney, “Design of deep learning algorithm for IoT application by image based recognition,” J. ISMAC, vol. 3, no. 03, pp. 276–290, 2021. https://doi.org/10.36548/jismac.2021.3.008

[8] H. Chen, Y. Wang, J. Guo, and D. Tao, “Vanillanet: the power of minimalism in deep learning,” Adv. Neural Inf. Process. Syst., vol. 36, pp. 7050–7064, 2023.

[9] M. N. Fekri, H. Patel, K. Grolinger, and V. Sharma, “Deep learning for load forecasting with smart meter data: Online Adaptive Recurrent Neural Network,” Appl. Energy, vol. 282, p. 116177, 2021. https://doi.org/10.1016/j.apenergy.2020.116177

[10] W. Albattah et al., “A novel deep learning method for detection and classification of plant diseases,” Complex Intell. Syst., pp. 1–18, 2022. https://doi.org/10.1007/s40747-021-00536-1

[11] S. F. Ahmed et al., “Deep learning modelling techniques: current progress, applications, advantages, and challenges,” Artif. Intell. Rev., vol. 56, no. 11, pp. 13521–13617, 2023. https://doi.org/10.1007/s10462-023-10466-8

[12] G. Menghani, “Efficient deep learning: A survey on making deep learning models smaller, faster, and better,” ACM Comput. Surv., vol. 55, no. 12, pp. 1–37, 2023. https://doi.org/10.1145/3578938

[13] A. Haridasan, J. Thomas, and E. D. Raj, “Deep learning system for paddy plant disease detection and classification,” Environ. Monit. Assess., vol. 195, no. 1, p. 120, 2023. https://doi.org/10.1007/s10661-022-10656-x

[14] A. K. Sharma et al., “Classification of Indian classical music with time-series matching deep learning approach,” IEEE Access, vol. 9, pp. 102041–102052, 2021. https://doi.org/10.1109/ACCESS.2021.3093911

[15] S. Fresca and A. Manzoni, “POD-DL-ROM: Enhancing deep learning-based reduced order models for nonlinear parametrized PDEs by proper orthogonal decomposition,” Comput. Methods Appl. Mech. Eng., vol. 388, p. 114181, 2022. https://doi.org/10.1016/j.cma.2021.114181

[16] X. Chen et al., “Recent advances and clinical applications of deep learning in medical image analysis,” Med. Image Anal., vol. 79, p. 102444, 2022. https://doi.org/10.1016/j.media.2022.102444

[17] Y. Zhang and Y. F. Li, “Prognostics and health management of Lithium-ion battery using deep learning methods: A review,” Renew. Sustain. Energy Rev., vol. 161, p. 112282, 2022. https://doi.org/10.1016/j.rser.2022.112282

[18] A. Mohan, A. K. Singh, B. Kumar, and R. Dwivedi, “Review on remote sensing methods for landslide detection using machine and deep learning,” Trans. Emerg. Telecommun. Technol., vol. 32, no. 7, e3998, 2021. https://doi.org/10.1002/ett.3998

[19] M. Hakim et al., “A systematic review of rolling bearing fault diagnoses based on deep learning and transfer learning,” Ain Shams Eng. J., vol. 14, no. 4, p. 101945, 2023. https://doi.org/10.1016/j.asej.2022.101945

[20] C. C. Ukwuoma et al., “Recent advancements in fruit detection and classification using deep learning techniques,” Math. Probl. Eng., vol. 2022, no. 1, p. 9210947, 2022. https://doi.org/10.1155/2022/9210947

[21] R. Archana and P. E. Jeevaraj, “Deep learning models for digital image processing: a review,” Artif. Intell. Rev., vol. 57, no. 1, p. 11, 2024. https://doi.org/10.1007/s10462-023-10631-z

[22] H. A. Helaly, M. Badawy, and A. Y. Haikal, “Deep learning approach for early detection of Alzheimer’s disease,” Cogn. Comput., vol. 14, no. 5, pp. 1711–1727, 2022. https://doi.org/10.1007/s12559-021-09946-2

[23] Y. Matsuzaka and Y. Uesawa, “A molecular image-based novel QSAR approach: deepsnap-deep learning,” Curr. Issues Mol. Biol., vol. 42, no. 1, pp. 455–472, 2021. https://doi.org/10.21775/cimb.042.455

[24] S. Singh and A. Mahmood, “The NLP cookbook: modern recipes for transformer based deep learning architectures,” IEEE Access, vol. 9, pp. 68675–68702, 2021. https://doi.org/10.1109/ACCESS.2021.3077350

[25] T. L. Chaunzwa et al., “Deep learning classification of lung cancer histology using CT images,” Sci. Rep., vol. 11, no. 1, pp. 1–12, 2021. https://doi.org/10.1038/s41598-021-84630-x

[26] H. Kaur, S. U. Ahsaan, B. Alankar, and V. Chang, “A proposed sentiment analysis deep learning algorithm for analyzing COVID-19 tweets,” Inf. Syst. Front., vol. 23, no. 6, pp. 1417–1429, 2021. https://doi.org/10.1007/s10796-021-10135-7

[27] C. Dimas, V. Alimisis, N. Uzunoglu, and P. P. Sotiriadis, “Advances in electrical impedance tomography inverse problem solution methods,” IEEE Access, vol. 12, pp. 47797–47829, 2024. https://doi.org/10.1109/ACCESS.2024.3382939

[28] U. S. Rao et al., “Deep learning precision farming: grapes and mango leaf disease detection by transfer learning,” Glob. Transit. Proc., vol. 2, no. 2, pp. 535–544, 2021. https://doi.org/10.1016/j.gltp.2021.08.002

[29] G. Roeder, L. Metz, and D. Kingma, “On linear identifiability of learned representations,” in Proc. Int. Conf. Mach. Learn. (ICML), pp. 9030–9039, Jul. 2021.

[30] S. Yu and J. Ma, “Deep learning for geophysics: Current and future trends,” Rev. Geophys., vol. 59, no. 3, e2021RG000742, 2021. https://doi.org/10.1029/2021RG000742

[31] A. Gasparin, S. Lukovic, and C. Alippi, “Deep learning for time series forecasting: The electric load case,” CAAI Trans. Intell. Technol., vol. 7, no. 1, pp. 1–25, 2022. https://doi.org/10.1049/cit2.12060

[32] I. H. Sarker, “Deep cybersecurity: a comprehensive overview from neural network and deep learning perspective,” SN Comput. Sci., vol. 2, no. 3, p. 154, 2021. https://doi.org/10.1007/s42979-021-00535-6

[33] M. Li et al., “A deep learning method of water body extraction from high resolution remote sensing images with multisensors,” IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens., vol. 14, pp. 3120–3132, 2021. https://doi.org/10.1109/JSTARS.2021.3060769

[34] M. S. Ismail and R. Peeters, “A connection between von Neumann-Morgenstern expected utility and symmetric potential games,” Theory Decis., pp. 1–14, 2024. https://doi.org/10.2139/ssrn.4615620

[35] R. Van Den Brink and A. Rusinowska, “Degree centrality, von Neumann–Morgenstern expected utility and externalities in networks,” Eur. J. Oper. Res., vol. 319, no. 2, pp. 669–677, 2024. https://doi.org/10.1016/j.ejor.2024.06.042

[36] J. Lopez-Wild, “A computable von Neumann-Morgenstern representation theorem,” Synthese, vol. 205, no. 5, p. 182, 2025. https://doi.org/10.1007/s11229-025-04980-1

[37] N. N. Osipov, “Von Neumann–Morgenstern Axioms of Rationality and Inequalities in Analysis,” J. Math. Sci., vol. 285, no. 1, pp. 142–153, 2024. https://doi.org/10.1007/s10958-024-07439-9

[38] N. T. Wilcox, “Utility measurement: Some contemporary concerns,” in A Modern Guide to Philosophy of Economics, Edward Elgar, pp. 14–27, 2021. https://doi.org/10.4337/9781788974462.00007

[39] J. C. Francis, “Harry Markowitz’s contributions to utility theory,” Ann. Oper. Res., vol. 346, no. 1, pp. 113–125, 2025. https://doi.org/10.1007/s10479-024-06210-2

[40] J. C. Francis, “Reformulating prospect theory to become a von Neumann–Morgenstern theory,” Rev. Quant. Finance Account., vol. 56, no. 3, pp. 965–985, 2021. https://doi.org/10.1007/s11156-020-00915-8

[41] J. GarcÍa Cabello, “A New Decision Making Method for Selection of Optimal Data Using the Von Neumann-Morgenstern Theorem,” Informatica, vol. 34, no. 4, pp. 771–794, 2023. https://doi.org/10.15388/23-INFOR530

[42] B. Hu and F. Yuan, "Utility-Probability Duality of Neural Networks," *arXiv preprint arXiv:2305.14859*, May 2023.

How to cite this article::

Zhang, J. (2025). A Novel Utility-Based Nonlinear Mapping Mechanism for Enhancing Linear Layers in Neural Networks . EPISTEMUS, 19(38), e3818442. https://doi.org/10.36790/epistemus.v19i38.442