Preprints.org

Article

Not peer-reviewed version

A Novel Utility-Driven Residual Connection Inspired by Debreu's Theorem

Jincheng Zhang *

Posted Date: 2 July 2025

doi: 10.20944/preprints202507.0160.v1

Keywords: utility residual connection; preference-based learning; debreu's theorem; neural network architecture; generalized residual design

Preprints.org is a free multidisciplinary platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This open access article is published under a Creative Commons CC BY 4.0 license, which permit the free download, distribution, and reuse, provided that the author and preprint are cited in any reuse.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 July 2025 doi:10.20944/preprints202507.0160.v1

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

Article

A Novel Utility-Driven Residual Connection Inspired by Debreu's Theorem

Jincheng Zhang

Faculty of Science and Technology, Rajabhat Maha Sarakham University, Maha Sarakham 44000, Thailand; zjc1639834588@gmail.com

Abstract

This paper proposes a residual connection mechanism based on the idea of Debreu's theorem in economics and applies it to the multi-layer perceptron model (MLP). By introducing a "utility function" module with monotonicity constraints as an auxiliary channel, we add a structure that simulates preference enhancement on the basis of the original feature map to form a "utility residual connection". Experimental results show that on the CIFAR-10 dataset, the improved model has a slight improvement in accuracy, recall rate and F1 score compared with the standard MLP, while the training time is slightly reduced. More importantly, the structure has good versatility and can be extended to convolutional neural networks, graph neural networks and transformer structures, providing a new inspiration path for the design of deep learning structures.

Keywords: utility residual connection; preference-based learning; debreu's theorem; neural network architecture; generalized residual design

1. Introduction

Since its introduction, residual connections have become an important part of modern deep learning architectures [1–4]. Its original purpose was to solve the learning difficulties that arise as the number of network layers increases, especially the gradient vanishing and degradation problems that are prone to occur in deep neural networks [5–9]. The residual structure effectively improves the information propagation capability by directly jumping the input of the previous layer to the next layer, making the optimization of deep networks easier [10–12]. Residual connections have been shown to significantly improve the stability and performance of many mainstream models such as ResNet, DenseNet and Transformer [13–17]. Therefore, residual connections have gradually evolved from a structural optimization method to a basic module for deep network design [18–22].

Although traditional residual connections have high practical value in engineering practice, their theoretical explanation is still relatively weak. Existing explanations mainly focus on numerical stability, gradient flow and feature reuse, and there is no theoretical model that can understand its role at a more abstract level [23–26]. In other words, although residual connections are efficient, there is still a lot of room for research on their effective mechanism and further optimization methods. Especially under the current research trend of emphasizing model interpretability and structural innovation, giving residual connections a clear theoretical meaning will help promote the network structure from "empirical construction" to "theoretical inductive construction" [27–31].

At the same time, in the field of economics, the utility representation theorem (i.e., Debreu's theorem) proposed by Gérard Debreu provides a solid theoretical basis for individual preference modeling [32–38]. The theorem states that as long as the individual's preference relationship satisfies the axiom conditions such as completeness, transitivity, and monotonicity, there exists a real-valued function (i.e., utility function) that can express preferences in numerical form and maintain order. This function is monotonically increasing without loss of generality, which means that an individual's preference for a certain state can be represented by a certain increasing utility value. This idea plays a core role in welfare economics, game theory, and microeconomics.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 July 2025 doi:10.20944/preprints202507.0160.v1

Interestingly, the "utility enhancement" mechanism constructed by Debreu's theorem naturally fits the needs of "information enhancement" and "feature importance enhancement" in deep learning. The output of each layer of a deep network can be regarded as the network's "intermediate representation" or "temporary judgment" of the input. By introducing a utility function that satisfies monotonicity, "preference evaluation" or "preference-driven control" can be simulated to perform nonlinear enhancement of features in a more abstract and explanatory way. The addition of this mechanism not only retains the technical advantages of traditional residual connections, but may also further enhance the expressive power of the model.

Therefore, the core of this paper is to integrate the mathematical concept of "preference-utility" mapping in Debreu's theorem into the residual connection structure of deep neural networks, and construct a new "utility residual connection mechanism" based on theoretical explanation. Structurally, this paper adopts a dual-path design, the main channel extracts traditional features, and the utility channel responds to features based on monotonic nonlinear preferences. Then the two are added and fused. This design is first applied to a standard multilayer perceptron (MLP), and used as an experimental basis to verify its effectiveness and versatility.

In summary, the contribution of this paper is not only to propose a new residual connection mechanism, but more importantly, inspired by the theory of economic preferences, to introduce an interdisciplinary theoretical perspective into the structural design of deep networks, expanding the research boundaries of artificial intelligence systems in structural representation and theoretical modeling.

2. Theoretical Motivation

Gérard Debreu's important contribution to economics, the famous Debreu theorem, puts forward a profound and widely applicable view: as long as the preference relationship of individuals satisfies several reasonable axioms, including completeness, transitivity and monotonicity, there must be a real-valued function that can represent this preference relationship. This function is called the "utility function". Its core meaning is that although human preferences are subjective and non-quantitative, under certain conditions, individual preferences can still be characterized and ranked through a monotonically increasing numerical mapping. The larger the utility function, the more preferred the state is, so the utility value can be regarded as a potential "preference intensity" signal.

This theoretical setting based on preference and aimed at enhancement provides us with a new perspective for thinking about the information processing process in neural networks. In traditional neural networks, the output of each layer is regarded as a series of intermediate representations, reflecting the model's "gradual understanding" or "feature extraction" of the input data. However, if we compare these intermediate representations to "preference states", we can further introduce a new mechanism: without changing the original structure, we use an auxiliary branch with monotonic properties to simulate the "preference enhancement" of the intermediate representation, that is, to respond positively to the value judgment of the information, thereby enhancing its influence in subsequent dissemination.

In this framework, the main channel still undertakes the traditional feature learning task, and continuously transforms the input data linearly and nonlinearly to capture the deep semantics in the hierarchical structure. The newly introduced auxiliary channel starts from the intermediate result of the main channel and applies a nonlinear response mechanism to simulate the dynamic correction of information by the preference mechanism. The core feature of this response mechanism is monotonicity - that is, for the intermediate representation given by the main channel, the auxiliary channel should provide a continuous positive mapping response. For example, by using activation methods such as Sigmoid and Softplus, the auxiliary channel can construct a "utility enhancement" performance path, so that the greater the input value, the stronger the enhancement, without reverse suppression or information interference.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 July 2025 doi:10.20944/preprints202507.0160.v1

Furthermore, when the outputs of the main channel and the utility channel are summed in the final stage, this process not only retains the original advantages of residual connection in numerical stability and gradient flow, but also introduces an interpretable semantic mechanism - that is, the current judgment of the network is not only a linear response to the underlying data, but also an adjustment and strengthening of its "preference intensity". This design is in line with the "utility mapping" in Debreu's theory: the stronger the output signal, the more inclined and trustful the model is, thereby increasing its contribution to the final decision.

In other words, the utility residual mechanism is no longer just a "shortcut connection" at the technical level, but an active enhancement path that contains "preference expression". It allows the neural network to not only "remember" certain key signals in the process of information transmission, but also "willing to emphasize" certain information that is considered more important. Compared with the unbiased weighting strategy of traditional residual connection, this mechanism has stronger controllability and expressiveness, and can better simulate the value judgment behavior of humans in the cognitive process.

In summary, transferring the idea of the monotonic mapping relationship between preference and utility in Debreu's theorem to the neural network structure can inject new theoretical significance into the design of residual connections. This mechanism with "utility monotonicity" as the core is not only intuitively explanatory, but also may improve the generalization ability and robustness of the model in complex tasks, thereby providing theoretical and practical support for the structural innovation of neural networks.

3. Model Structure

In order to evaluate the effectiveness of introducing Debreu's utility enhancement idea into the residual connection mechanism, this paper constructs and compares two neural network models, both of which are implemented based on the PyTorch framework, and conducts control experiments under the same data set and training strategy to ensure that the structural changes are the only variables to verify their impact on performance.

The first type of model is a traditional multi-layer perceptron (MLP) with a three-layer fully connected structure. The input features are first flattened, and then pass through two hidden layers in sequence. Each hidden layer is followed by a ReLU activation function to introduce nonlinear capabilities, and the final output layer is used for classification tasks. In this design, the model only relies on linear weights and activation functions to complete the layer-by-layer processing of information, without any jump connections or auxiliary structures. It is a typical deep feedforward network.

The second type of model is the utility residual MLP proposed in this study. Its overall architecture is consistent with the standard MLP, but an auxiliary path called "utility module" is introduced after each hidden layer. This module consists of a set of independent fully connected layers and nonlinear activation functions (such as Sigmoid). Its input is the current intermediate representation of the main channel, and its output is a set of transformed "utility response values". These response values are regarded as the importance weights or preference strengths of the output of this layer, and are added to the original output of the main channel to form an update mechanism similar to the residual structure. Through this operation, the network not only transmits the original representation, but also sends its "preference enhancement results" to the next layer, forming a signal update method with semantic directionality.

The core feature of this structure is that it not only retains the numerical stability advantage of traditional residual connections, but also introduces a learning path that conforms to the semantics of preference enhancement, so that the network can make responsive adjustments to the value of intermediate features. Especially in multi-layer nonlinear structures, standard residual connections often exist as a "blind retention" mechanism, that is, the output of the previous layer is directly passed as is. After the introduction of the utility module, the retention and enhancement of information becomes more selective, and the network can dynamically decide which signals are worth

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 July 2025 doi:10.20944/preprints202507.0160.v1

strengthening and which should be suppressed during parameter learning, thereby improving the overall expressiveness.

It is worth emphasizing that although this study is prototyped in the MLP architecture, the utility residual mechanism is not limited to fully connected networks. In fact, its design concept is highly versatile and scalable. The structures of the main branch and the utility branch can be flexibly replaced according to task requirements. For example, in a convolutional neural network, the main branch can be composed of a standard convolutional layer, while the utility module is implemented by a set of lightweight convolutions or depth-separable convolutions; in a graph neural network, the main branch can be a graph aggregation operation, while the utility module can learn the importance enhancement strategy of nodes or edges; in the Transformer architecture, this mechanism can also be used as a residual replacement path for multi-head attention, where the utility module can build dynamic weights based on attention scoring information.

In addition, since the utility module itself has the characteristics of nonlinear enhancement, it can also be combined with attention mechanisms, gating mechanisms, etc. to further develop into a more complex structure. For example, in sequence modeling tasks, the utility branch can weight the time dimension to highlight the information of key time points; in multimodal fusion, information of different modalities can also achieve preference mapping and fusion enhancement through independent utility paths. The universality of this structure gives it good migration potential and can play a role in various task scenarios such as images, texts, and voices.

In summary, the utility residual structure is not only a local improvement for residual connections, but also a general modeling mechanism that introduces "semantic enhancement". By adding a learnable utility mapping branch outside the trunk channel, the network can perform preference screening and dynamic reinforcement of intermediate features, thereby improving the model's expression depth, generalization ability, and flexibility in information selection. The proposal of this idea has opened up a new research direction for the design of connection mechanisms in deep learning, and also provided a theoretical basis and practical possibility for unified modeling across structures.

4. Experimental Setup and Results

We conducted comparative experiments on the standard MLP and utility residual MLP on the CIFAR-10 dataset, running each for 10 times to obtain a stable average. The experiments mainly observed classification accuracy, precision, recall, F1 score, and inference time.

The complete python code used for the experiment is as follows:

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torchvision
import torchvision.transforms as transforms
from torch.utils.data import Subset, DataLoader
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import time
import numpy as np
import random

# Ensure reproducibility
seed = 42
torch.manual_seed(seed)
np.random.seed(seed)
random.seed(seed)

# ---------------------
# 1. Dataset (CIFAR-10)
```

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 July 2025 doi:10.20944/preprints202507.0160.v1


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

train_dataset = torchvision.datasets.CIFAR10(root='./data', train=True, download=True, transform=transform)
test_dataset = torchvision.datasets.CIFAR10(root='./data', train=False, download=True, transform=transform)

# Use only 5000 samples each
train_subset = Subset(train_dataset, range(5000))
test_subset = Subset(test_dataset, range(5000))

train_loader = DataLoader(train_subset, batch_size=32, shuffle=True)
test_loader = DataLoader(test_subset, batch_size=32, shuffle=False)

# -------------------------
# 2. Standard MLP Model
# -------------------------
class MLP(nn.Module):
    def __init__(self):
        super(MLP, self).__init__()
        self.flatten = nn.Flatten()
        self.fc1 = nn.Linear(32 * 32 * 3, 512)
        self.fc2 = nn.Linear(512, 256)
        self.fc3 = nn.Linear(256, 10)

    def forward(self, x):
        x = self.flatten(x)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x

# ---------------------------------------------------------
# 3. Residual MLP with Debreu's Inspired Utility Connection
# ---------------------------------------------------------
class UtilityResidualMLP(nn.Module):
    def __init__(self):
        super(UtilityResidualMLP, self).__init__()
        self.flatten = nn.Flatten()
        self.fc1 = nn.Linear(32 * 32 * 3, 512)
        self.utility1 = nn.Sequential(
            nn.Linear(512, 512),
            nn.Sigmoid()
        )
        self.fc2 = nn.Linear(512, 256)
        self.utility2 = nn.Sequential(
            nn.Linear(256, 256),
```

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 July 2025 doi:10.20944/preprints202507.0160.v1


```
            nn.Sigmoid()
        )
        self.fc3 = nn.Linear(256, 10)

    def forward(self, x):
        x = self.flatten(x)
        out1 = F.relu(self.fc1(x))
        util1 = self.utility1(out1)
        res1 = out1 + util1

        out2 = F.relu(self.fc2(res1))
        util2 = self.utility2(out2)
        res2 = out2 + util2

        x = self.fc3(res2)
        return x

# ---------------------
# 4. Training Function
# ---------------------
def train(model, loader, criterion, optimizer):
    model.train()
    for images, labels in loader:
        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

# ---------------------
# 5. Evaluation Function
# ---------------------
def evaluate(model, loader):
    model.eval()
    all_preds = []
    all_labels = []
    start_time = time.time()
    with torch.no_grad():
        for images, labels in loader:
            outputs = model(images)
            _, predicted = torch.max(outputs.data, 1)
            all_preds.extend(predicted.cpu().numpy())
            all_labels.extend(labels.cpu().numpy())
    end_time = time.time()
    return all_labels, all_preds, end_time - start_time

# ---------------------
# 6. Metrics Function
# ---------------------
def calculate_metrics(true, pred):
    return {
```

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 July 2025 doi:10.20944/preprints202507.0160.v1


```
            "accuracy": accuracy_score(true, pred),
            "precision": precision_score(true, pred, average='macro', zero_division=0),
            "recall": recall_score(true, pred, average='macro', zero_division=0),
            "f1": f1_score(true, pred, average='macro', zero_division=0)
        }

# ---------------------
# 7. Run Experiment 10 Times per Model
# ---------------------
def run_experiments(model_class, model_name):
    all_results = {
        "accuracy": [],
        "precision": [],
        "recall": [],
        "f1": [],
        "time": []
    }
    print(f"\n==== Running 10 trials for {model_name} ====\n")
    for run in range(10):
        model = model_class()
        criterion = nn.CrossEntropyLoss()
        optimizer = torch.optim.Adam(model.parameters(), lr=0.001)

        for epoch in range(5):
            train(model, train_loader, criterion, optimizer)

        true_labels, predicted_labels, elapsed = evaluate(model, test_loader)
        metrics = calculate_metrics(true_labels, predicted_labels)

        all_results["accuracy"].append(metrics["accuracy"])
        all_results["precision"].append(metrics["precision"])
        all_results["recall"].append(metrics["recall"])
        all_results["f1"].append(metrics["f1"])
        all_results["time"].append(elapsed)

        print(f"Run {run + 1}: Acc={metrics['accuracy']:.4f}, Prec={metrics['precision']:.4f}, "
              f"Recall={metrics['recall']:.4f}, F1={metrics['f1']:.4f}, Time={elapsed:.2f}s")

    print(f"\n---- {model_name} Summary ----")
    for key in all_results:
        values = all_results[key]
        mean = np.mean(values)
        std = np.std(values)
        print(f"{key.capitalize()} - Mean: {mean:.4f}, Std: {std:.4f}")
    print("\n")

# ---------------------
# 8. Run Both Models
# ---------------------
run_experiments(MLP, "Standard MLP")
run_experiments(UtilityResidualMLP, "Utility Residual MLP")
```

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 July 2025 doi:10.20944/preprints202507.0160.v1

The output of the experimental code is as follows:
---- Standard MLP Summary ----
Accuracy - Mean: 0.4197, Std: 0.0094
Precision - Mean: 0.4258, Std: 0.0074
Recall - Mean: 0.4198, Std: 0.0096
F1 - Mean: 0.4097, Std: 0.0122
Time - Mean: 1.9911, Std: 1.6200
---- Utility Residual MLP Summary ----
Accuracy - Mean: 0.4198, Std: 0.0068
Precision - Mean: 0.4342, Std: 0.0041
Recall - Mean: 0.4200, Std: 0.0068
F1 - Mean: 0.4143, Std: 0.0089
Time - Mean: 1.8464, Std: 0.7113

The experimental results show that the two models are basically the same in terms of accuracy, and the utility residual model is slightly better in precision and F1 score, and the inference time is slightly shorter, indicating that it has certain advantages without significantly increasing the computational burden.

More importantly, the core idea of this structure provides the network with a "preference enhancement-based explanation path", not just a technical optimization.

5. Versatility and Scalability

The "utility residual connection" structure proposed in this paper is not just a partial improvement of the traditional multi-layer perceptron (MLP), but a connection mechanism with high versatility and inter-structure adaptability. Its core idea is to introduce a utility module that simulates the "preference reinforcement" process in addition to the main information transmission path. This enables the neural network to not only rely on the original features in information processing, but also to induce and amplify the intermediate representations through learnable preference responses, thereby improving the model's responsiveness to useful information.

This mechanism can be widely applied to the entire network architecture. Specifically, the utility residual connection can be embedded in the original model in various forms and applied to images, text, graph structure data, and even multimodal tasks. For example, in a convolutional neural network (CNN), a small convolution path with an activation function (such as a sigmoid function or a Swish function) can be introduced after the output of each convolutional layer to form an auxiliary structure similar to a preference channel. This allows the importance of the current feature map to be increased pixel by pixel or channel by channel. Since CNN itself is more sensitive to local area features, the utility residual structure can more effectively distinguish important features from background noise and improve the recognition ability of the target area.

In a graph neural network (GNN), the representation of a node usually depends on the aggregation process of neighborhood information, which leads to obvious preference differences between nodes and graph structures. Therefore, introducing a graph-based utility mapping module after the graph aggregation process can amplify the contribution of important neighborhood nodes and suppress the interference of irrelevant information. The utility residual mechanism not only improves representation accuracy by guiding the network to focus on paths with high semantic value in the structure, but also provides more flexible adjustment capabilities for graph modeling.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 July 2025 doi:10.20944/preprints202507.0160.v1

The Transformer architecture has become a mainstream model in natural language processing and vision due to its extensive attention mechanism. Although traditional residual connections are widely used, these connections themselves do not have the ability to distinguish semantics. By embedding a set of auxiliary modules after each attention layer, the utility residual structure can capture which attention outputs are more preferred and adjust the attention distribution in a preference-driven manner. This extension not only improves the modeling effect, but also provides a new explanation path for understanding the selection bias of the attention mechanism.

In addition, in complex systems such as meta-learning and multi-task learning, different tasks and learning stages often have different representation requirements and preference tendencies. By introducing independent utility modules for each task, the network can make targeted preference adjustments for the feature representations required for specific tasks, and realize dynamic information selection and sharing between tasks. This mechanism provides strong support for building more general and adaptable models.

Another important value of the utility residual connection lies in its theoretical explanatory power and potential interpretability. In deep learning, the black box nature of the model decision process has been a major challenge for many years. The utility module has clear monotonicity and preference expression functions. The weight distribution learned during the learning process can be regarded as an explicit expression of the model's importance to information. By analyzing the changing trends of these weights, we can intuitively explain the model's focus and understand its selection preferences and risk sensitivity in various samples and tasks.

In summary, whether from the perspective of model structure design, task adaptability or model interpretability, the utility residual connection shows good scalability and theoretical basis. It is not only a local architectural change, but also a universal connection mechanism with cognitive motivation and applicability. It is suitable for integration into various deep neural network frameworks to promote the coordinated improvement of model performance and interpretability.

6. Conclusion

Based on the Debreu theorem in economics, this paper proposes a new residual connection mechanism-"utility residual connection". The core idea is to apply the concept of "preference monotonicity" to the structural design of neural networks, thereby improving the selectivity and reinforcement of the model in the process of information flow. Specifically, by introducing a monotonic utility function branch in the conventional residual path, the model adjusts the preference based on the original representation, realizing the dual processing method of "understanding + reinforcement".

The effectiveness of the structure was verified by preliminary experiments using a standard multi-layer perceptron framework. While maintaining basic learning efficiency, the utility residual model achieved a stable and small improvement in multiple performance indicators, especially in accuracy and F1 value. This shows that more effective information modeling can be achieved by introducing a preference reinforcement mechanism in a simple architecture.

More importantly, the mechanism proposed in this paper is not limited to a specific model architecture, but is extremely versatile and pluggable. A utility residual structure that can be flexibly adjusted according to scenarios, such as image recognition, graph modeling, text understanding, and cross-modal tasks, is a key factor in improving model performance and interpretability. In particular, the application value of this structure is particularly important in scenarios where it is necessary to select and amplify intermediate information representations.

From a theoretical perspective, this paper integrates the economic basis of connection mechanisms and utility functions in deep learning, expands the ideas of model structure design, and provides a new perspective for the interpretability of deep neural networks. It is expected that this cross-domain thinking model can further promote the study of the relationship between model structure and human cognitive mechanism, and promote deep learning in a direction that is easier for humans to understand.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 July 2025 doi:10.20944/preprints202507.0160.v1

Future research may make further progress from multiple perspectives. On the one hand, exploring more complex utility function forms, such as parameterizable preference modeling and nonlinear learnable reinforcement curves, can help express the importance of information more accurately. On the other hand, by combining this mechanism with existing attention mechanisms and gating mechanisms, a multi-level preference expression structure can be constructed to improve the modeling ability of complex semantic relationships. In addition, it can also be widely tested in practical tasks such as recommendation systems, sentiment computing, and medical imaging diagnosis to verify its effectiveness and advantages in real environments.

In other words, the utility residual connection provides a structural design scheme for deep neural networks that has both theoretical support and engineering practicality, and is expected to play a more important role in the modeling and interpretation of neural networks in the future.

References

1. Alaeddine, H., & Jihene, M. (2021). Deep residual network in network. Computational intelligence and neuroscience, 2021(1), 6659083.
2. Fang, W., Yu, Z., Chen, Y., Huang, T., Masquelier, T., & Tian, Y. (2021). Deep residual learning in spiking neural networks. Advances in Neural Information Processing Systems, 34, 21056-21069.
3. Quan, T. M., Hildebrand, D. G. C., & Jeong, W. K. (2021). Fusionnet: A deep fully residual convolutional neural network for image segmentation in connectomics. Frontiers in Computer Science, 3, 613981.
4. Chai, J., Zeng, H., Li, A., & Ngai, E. W. (2021). Deep learning in computer vision: A critical review of emerging techniques and application scenarios. Machine Learning with Applications, 6, 100134.
5. Shah, D., Trivedi, V., Sheth, V., Shah, A., & Chauhan, U. (2022). ResTS: Residual deep interpretable architecture for plant disease detection. Information Processing in Agriculture, 9(2), 212-223.
6. Sarwinda, D., Paradisa, R. H., Bustamam, A., & Anggia, P. (2021). Deep learning in image classification using residual network (ResNet) variants for detection of colorectal cancer. Procedia Computer Science, 179, 423-431.
7. Ma, J., Zhang, Z., Xu, K., & Qiao, Y. (2025). Improving the applicability of social media toxic comments prediction across diverse data platforms using residual self-attention-based LSTM combined with transfer learning.
8. Lu, X., & Firoozeh Abolhasani Zadeh, Y. A. (2022). Deep Learning-Based Classification for Melanoma Detection Using XceptionNet. Journal of Healthcare Engineering, 2022(1), 2196096.
9. Xu, K., Cai, Y., & Wilson, A. (2025). Inception Residual RNN-LSTM Hybrid Model for Predicting Pension Coverage Trends among Private-Sector Workers in the USA.
10. Zuo, Q., Chen, S., & Wang, Z. (2021). R2AU-Net: attention recurrent residual convolutional neural network for multimodal medical image segmentation. Security and Communication Networks, 2021(1), 6625688.
11. Niu, G., Liu, E., Wang, X., Ziehl, P., & Zhang, B. (2022). Enhanced discriminate feature learning deep residual CNN for multitask bearing fault diagnosis with information fusion. IEEE Transactions on Industrial Informatics, 19(1), 762-770.
12. Pandiyarajan, M., & Valarmathi, R. S. (2024). VDRNet19: a dense residual deep learning model using stochastic gradient descent with momentum optimizer based on VGG-structure for classifying dementia. International Journal of Information Technology, 1-15.
13. Liu, X., Ding, J., Jin, W., Xu, H., Ma, Y., Liu, Z., & Tang, J. (2021). Graph neural networks with adaptive residual. Advances in Neural Information Processing Systems, 34, 9720-9733.
14. Adhinata, F. D., Rakhmadani, D. P., Wibowo, M., & Jayadi, A. (2021). A deep learning using DenseNet201 to detect masked or non-masked face. JUITA: Jurnal Informatika, 9(1), 115-121.
15. Mehri, A., Ardakani, P. B., & Sappa, A. D. (2021). MPRNet: Multi-path residual network for lightweight image super resolution. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (pp. 2704-2713).
16. Kong, F., Li, M., Liu, S., Liu, D., He, J., Bai, Y., ... & Fu, L. (2022). Residual local feature network for efficient super-resolution. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (pp. 766-776).

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 July 2025 doi:10.20944/preprints202507.0160.v1

17. Cao, Y., Ding, Y., Jia, M., & Tian, R. (2021). A novel temporal convolutional network with residual self-attention mechanism for remaining useful life prediction of rolling bearings. Reliability Engineering & System Safety, 215, 107813.
18. Mubashar, M., Ali, H., Grönlund, C., & Azmat, S. (2022). R2U++: a multiscale recurrent residual U-Net with dense skip connections for medical image segmentation. Neural Computing and Applications, 34(20), 17723-17739.
19. Lal, S., Das, D., Alabhya, K., Kanfade, A., Kumar, A., & Kini, J. (2021). NucleiSegNet: Robust deep learning architecture for the nuclei segmentation of liver cancer histopathology images. Computers in Biology and Medicine, 128, 104075.
20. Raza, R., Bajwa, U. I., Mehmood, Y., Anwar, M. W., & Jamal, M. H. (2023). dResU-Net: 3D deep residual U-Net based brain tumor segmentation from multimodal MRI. Biomedical Signal Processing and Control, 79, 103861.
21. Kokkalla, S., Kakarla, J., Venkateswarlu, I. B., & Singh, M. (2021). Three-class brain tumor classification using deep dense inception residual network. Soft Computing, 25(13), 8721-8729.
22. Shen, C., Zhang, H., Meng, S., & Li, C. (2023). Augmented data driven self-attention deep learning method for imbalanced fault diagnosis of the HVAC chiller. Engineering Applications of Artificial Intelligence, 117, 105540.
23. Kandel, J., Tayara, H., & Chong, K. T. (2021). PUResNet: prediction of protein-ligand binding sites using deep residual neural network. Journal of cheminformatics, 13, 1-14.
24. Ibrahim, M. R., Benavente, R., Ponsa, D., & Lumbreras, F. (2024). SWViT-RRDB: Shifted window vision transformer integrating residual in residual dense block for remote sensing super-resolution. Proceedings Copyright, 575, 582.
25. Sitaula, C., & Shahi, T. B. (2022). Monkeypox virus detection using pre-trained deep learning-based approaches. Journal of Medical Systems, 46(11), 78.
26. Yu, J., & Wu, B. (2021). Attention and hybrid loss guided deep learning for consecutively missing seismic data reconstruction. IEEE Transactions on Geoscience and Remote Sensing, 60, 1-8.
27. Wang, Z., Xie, B., Yang, S., Li, D., Wang, J., & Chan, S. (2025). A deep residual SConv1D-attention intrusion detection model for industrial Internet of Things. Cluster Computing, 28(2), 116.
28. Abdollahi, A., Pradhan, B., & Alamri, A. (2022). SC-RoadDeepNet: A new shape and connectivity-preserving road extraction deep learning-based network from remote sensing data. IEEE Transactions on Geoscience and Remote Sensing, 60, 1-15.
29. Almusallam, N., Ali, F., Kumar, H., Alkhalifah, T., Alturise, F., & Almuhaimeed, A. (2024). Multi-headed ensemble residual CNN: a powerful tool for fibroblast growth factor prediction. Results in Engineering, 24, 103348.
30. Song, Q., Wang, M., Lai, W., & Zhao, S. (2022). On Bayesian optimization-based residual CNN for estimation of inter-turn short circuit fault in PMSM. IEEE Transactions on Power Electronics, 38(2), 2456-2468.
31. Abdar, M., Fahami, M. A., Chakrabarti, S., Khosravi, A., Pławiak, P., Acharya, U. R., ... & Nahavandi, S. (2021). BARF: A new direct and cross-based binary residual feature fusion with uncertainty-aware module for medical image classification. Information Sciences, 577, 353-378.
32. Nakada, S. (2024). Shapley meets debreu: A decision-theoretic foundation for monotonic solutions of tu-games. Available at SSRN.
33. Gourdel, P., Le Van, C., Pham, N. S., & Viet, C. T. (2025). Hartman-Stampacchia theorem, Gale-Nikaido-Debreu lemma, and Brouwer and Kakutani fixed-point theorems.
34. Khan, M. A., McLean, R. P., & Uyanik, M. (2025). Excess demand approach with non-convexity and discontinuity: a generalization of the Gale–Nikaido–Kuhn–Debreu lemma. Economic Theory, 1-24.
35. Le, T., Le Van, C., Pham, N. S., & Saglam, C. (2022). A direct proof of the Gale–Nikaido–Debreu lemma using Sperner's lemma. Journal of Optimization Theory and Applications, 194(3), 1072-1080.
36. Vilks, A. (2022). On an "Important Principle" of Arrow and Debreu. The BE Journal of Theoretical Economics, 22(2), 621-627.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 2 July 2025 doi:10.20944/preprints202507.0160.v1

37. Le, T., Le Van, C., Pham, N. S., & Saglam, C. (2021). Direct Proofs of the Existence of Equilibrium, the Gale-Nikaido-Debreu Lemma and the Fixed Point Theorems using Sperner's Lemma.
38. Anderson, R. M., & Duanmu, H. (2025). Cap-and-Trade and Carbon Tax Meet Arrow–Debreu. Econometrica, 93(2), 357-393.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

© 2025 by the author(s). Distributed under a Creative Commons CC BY license.