

AI Report

We predict this text is

Human Generated

AI Probability

0%

This number is the probability that the document is AI generated, not a percentage of AI text in the document.

Plagiarism



The plagiarism scan was not run for this document. Go to gptzero.me to check for plagiarism.

He_Deep_Residual_Learning_CVPR_2016_paper.pdf - 8/30/2026

This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.

Except for this watermark, it is identical to the version available on IEEE Xplore.

Deep Residual Learning for Image Recognition

Kaiming He

Xiangyu Zhang

Shaoqing Ren

Jian Sun

Microsoft Research

kahe v xiangz v shren jiansun microsoft com

Deeper neural networks are more difficult to train.

We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously.

We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions.

We provide comprehensive empirical evidence showing that these residual networks are easier to optimize, and can gain accuracy from considerably increased depth.

On the ImageNet dataset we evaluate residual nets with a depth of up to 152 layers—8× deeper than VGG nets [40] but still having lower complexity.

An ensemble of these residual nets achieves 3.57% error on the ImageNet test set.

This result won the 1st place on the ILSVRC 2015 classification task.

We also present analysis on CIFAR-10 with 100 and 1000 layers.

The depth of representations is of central importance for many visual recognition tasks.

Solely due to our extremely deep representations, we obtain a 28% relative improvement on the COCO object detection dataset.

Deep residual nets are foundations of our submissions to ILSVRC & COCO 2015 competitions¹, where we also won the 1st places on the tasks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation.

Deep convolutional neural networks [22, 21] have led to a series of breakthroughs for image classification [21, 49, 39].

Deep networks naturally integrate low/mid/high-level features [49] and classifiers in an end-to-end multi-layer fashion, and the “levels” of features can be enriched by the number of stacked layers (depth).

Recent evidence [40, 43] reveals that network depth is of crucial importance, and the leading results [40, 43, 12, 16] on the challenging ImageNet dataset [35] all exploit “very deep” [40] models, with a depth of sixteen [40] to thirty [16].

Many other non-trivial visual recognition tasks [7, 11, 6, 32, 27] have also

Figure 1.

Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer “plain” networks.

The deeper network has higher training error, and thus test error.

Similar phenomena on ImageNet is presented in Fig.

4.

greatly benefited from very deep models.

Driven by the significance of depth, a question arises: Is learning better networks as easy as stacking more layers?

An obstacle to answering this question was the notorious problem of vanishing/exploding gradients [14, 1, 8], which hamper convergence from the beginning.

This problem, however, has been largely addressed by normalized initialization [23, 8, 36, 12] and intermediate normalization layers [16], which enable networks with tens of layers to start converging for stochastic gradient descent (SGD) with backpropagation [22].

When deeper networks are able to start converging, a degradation problem has been exposed: with the network depth increasing, accuracy gets saturated (which might be unsurprising) and then degrades rapidly.

Unexpectedly, such degradation is not caused by overfitting, and adding more layers to a suitably deep model leads to higher training error, as reported in [10, 41] and thoroughly verified by our experiments.

Fig.

1 shows a typical example.

The degradation (of training accuracy) indicates that not all systems are similarly easy to optimize. Let us consider a shallower architecture and its deeper counterpart that adds more layers onto it. There exists a solution by construction to the deeper model: the added layers are identity mapping, and the other layers are copied from the learned shallower model. The existence of this constructed solution indicates that a deeper model should produce no higher training error than its shallower counterpart. But experiments show that our current solvers on hand are unable to find solutions that

1http image net org challenges LSVRC 2015 and http mscoco org dataset detections challenge2015.

770

Figure 2.
Residual learning: a building block.

are comparably good or better than the constructed solution (or unable to do so in feasible time).

In this paper, we address the degradation problem by introducing a deep residual learning framework. Instead of hoping each few stacked layers directly fit a desired underlying mapping, we explicitly let these layers fit a residual mapping.

Formally, denoting the desired underlying mapping as $\mathcal{H}(\mathbf{x})$, we let the stacked nonlinear layers fit another mapping of $\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}$.

The original mapping is recast into $\mathcal{F}(\mathbf{x}) + \mathbf{x}$.

We hypothesize that it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping.

To the extreme, if an identity mapping were optimal, it would be easier to push the residual to zero than to fit an identity mapping by a stack of nonlinear layers.

The formulation of $\mathcal{F}(\mathbf{x}) + \mathbf{x}$ can be realized by feedforward neural networks with "shortcut connections" (Fig.

Shortcut connections [2, 33, 48] are those skipping one or more layers.

In our case, the shortcut connections simply perform identity mapping, and their outputs are added to the outputs of the stacked layers (Fig.

Identity shortcut connections add neither extra parameter nor computational complexity.

The entire network can still be trained end-to-end by SGD with backpropagation, and can be easily implemented using common libraries (e.g., Caffe [19]) without modifying the solvers.

We present comprehensive experiments on ImageNet [35] to show the degradation problem and evaluate our method.

We show that: 1) Our extremely deep residual nets are easy to optimize, but the counterpart "plain" nets (that simply stack layers) exhibit higher training error when the depth increases; 2) Our deep residual nets can easily enjoy accuracy gains from greatly increased depth, producing results substantially better than previous networks.

Similar phenomena are also shown on the CIFAR-10 set [20], suggesting that the optimization difficulties and the effects of our method are not just akin to a particular dataset.

We present successfully trained models on this dataset with over 100 layers, and explore models with over 1000 layers.

On the ImageNet classification dataset [35], we obtain excellent results by extremely deep residual nets. Our 152-layer residual net is the deepest network ever presented on ImageNet, while still having lower complexity than VGG nets [40].

Our ensemble has 3.57% top-5 error on the

ImageNet test set, and won the 1st place in the ILSVRC 2015 classification competition.

The extremely deep representations also have excellent generalization performance on other recognition tasks, and lead us to further win the 1st places on: ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation in ILSVRC & COCO 2015 competitions.

This strong evidence shows that the residual learning principle is generic, and we expect that it is applicable in other vision and non-vision problems.

Residual Representations.

In image recognition, VLAD [18] is a representation that encodes by the residual vectors with respect to a dictionary, and Fisher Vector [30] can be formulated as a probabilistic version [18] of VLAD.

Both of them are powerful shallow representations for image retrieval and classification [4, 47].

For vector quantization, encoding residual vectors [17] is shown to be more effective than encoding original vectors.

In low-level vision and computer graphics, for solving Partial Differential Equations (PDEs), the widely used Multigrid method [3] reformulates the system as subproblems at multiple scales, where each subproblem is responsible for the residual solution between a coarser and a finer scale.

An alternative to Multigrid is hierarchical basis preconditioning [44, 45], which relies on variables that represent residual vectors between two scales.

It has been shown [3, 44, 45] that these solvers converge much faster than standard solvers that are unaware of the residual nature of the solutions.

These methods suggest that a good reformulation or preconditioning can simplify the optimization.

Shortcut Connections.

Practices and theories that lead to shortcut connections [2, 33, 48] have been studied for a long time.

An early practice of training multi-layer perceptrons (MLPs) is to add a linear layer connected from the network input to the output [33, 48].

In [43, 24], a few intermediate layers are directly connected to auxiliary classifiers for addressing vanishing/exploding gradients.

The papers of [38, 37, 31, 46] propose methods for centering layer responses, gradients, and propagated errors, implemented by shortcut connections.

In [43], an "inception" layer is composed of a shortcut branch and a few deeper branches.

Concurrent with our work, "highway networks" [41, 42] present shortcut connections with gating functions [15]. These gates are data-dependent and have parameters, in contrast to our identity shortcuts that are parameter-free.

When a gated shortcut is "closed" (approaching zero), the layers in highway networks represent non-residual functions.

On the contrary, our formulation always learns residual functions; our identity shortcuts are never closed, and all information is always passed through, with additional residual functions to be learned.

In addition, high-

771

way networks have not demonstrated accuracy gains with extremely increased depth (e.g., over 100 layers).

Let us consider $\mathcal{H}(\mathbf{x})$ as an underl

High Human Impact ● ● ● ● ● ● High AI Impact

FAQs

What is GPTZero?

GPTZero is the leading AI detector for checking whether a document was written by a large language model such as ChatGPT. GPTZero detects AI on sentence, paragraph, and document level. Our model was trained on a large, diverse corpus of human-written and AI-generated text with support for English, Spanish, French, German, and other languages. To date, GPTZero has served over 17 million users around the world, and works with over 100 organizations in education, hiring, publishing, legal, and more.

When should I use GPTZero?

Our users have seen the use of AI-generated text proliferate into education, certification, hiring and recruitment, social writing platforms, disinformation, and beyond. We've created GPTZero as a tool to highlight the possible use of AI in writing text. In particular, we focus on classifying AI use in prose. Overall, our classifier is intended to be used to flag situations in which a conversation can be started (for example, between educators and students) to drive further inquiry and spread awareness of the risks of using AI in written work.

Does GPTZero only detect ChatGPT outputs?

No, GPTZero works robustly across a range of AI language models, including but not limited to ChatGPT, GPT-5, GPT-4, GPT-3, Gemini, Claude, and AI services based on those models.

What are the limitations of the classifier?

The nature of AI-generated content is changing constantly. As such, these results should not be used to punish students. We recommend educators to use our behind-the-scene [Writing Reports](#) as part of a holistic assessment of student work. There always exist edge cases with both instances where AI is classified as human, and human is classified as AI. Instead, we recommend educators take approaches that give students the opportunity to demonstrate their understanding in a controlled environment and craft assignments that cannot be solved with AI. Our classifier is not trained to identify AI-generated text after it has been heavily modified after generation (although we estimate this is a minority of the uses for AI-generation at the moment). Currently, our classifier can sometimes flag other machine-generated or highly procedural text as AI-generated, and as such, should be used on more descriptive portions of text.

I'm an educator who has found AI-generated text by my students. What do I do?

Firstly, at GPTZero, we don't believe that any AI detector is perfect. There always exist edge cases with both instances where AI is classified as human, and human is classified as AI. Nonetheless, we recommend that educators can do the following when they get a positive detection: Ask students to demonstrate their understanding in a controlled environment, whether that is through an in-person assessment, or through an editor that can track their edit history (for instance, using our [Writing Reports](#) through Google Docs). Check out our list of [several recommendations](#) on types of assignments that are difficult to solve with AI.

Ask the student if they can produce artifacts of their writing process, whether it is drafts, revision histories, or brainstorming notes. For example, if the editor they used to write the text has an edit history (such as Google Docs), and it was typed out with several edits over a reasonable period of time, it is likely the student work is authentic. You can use GPTZero's Writing Reports to replay the student's writing process, and view signals that indicate the authenticity of the work.

See if there is a history of AI-generated text in the student's work. We recommend looking for a long-term pattern of AI use, as opposed to a single instance, in order to determine whether the student is using AI.