УДК 539.3

APPLICATION OF THE LOW-RANK ADAPTATION METHOD ON THE EXAMPLE OF FINE-TUNING A LATENT DIFFUSION MODEL

H.M. Ivanchenko[1],
Doctor of Technical Sciences, Professor

G.V. Getun[1],
Candidate of Technical Sciences, Associate Professor

I.O. Skliarov[1],
Candidate of Technical Sciences, Associate Professor

A.V. Solomin[2],
Candidate of Physical and Mathematical Sciences, Associate Professor

S.Y. Getun[1],
Postgraduate Student

[1] *Kyiv National University of Construction and Architecture, Kyiv*
[2] *National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv*

DOI: 10.32347/2410-2547.2025.114.299-310

This article explores the Low-Rank Adaptation (LoRA) method, a fast fine-tuning technique for large-parameter neural networks, and its potential application in architecture, construction, and structural mechanics. The study focuses on applying LoRA to fine-tune a Latent Diffusion Model (LDM) for generating images of buildings in various architectural styles. The mathematical foundation of diffusion models, the structure of LDMs, and key aspects of implementing fast fine-tuning of large neural networks on specialized data are discussed. The importance of a validation dataset in preventing overfitting and determining the optimal stopping point for training is also demonstrated. The authors aim to promote the use of neural networks in addressing challenges in structural mechanics, construction, and architecture, in addition to developing practical AI-based tools.

Keywords: Low-Rank Adaptation, Fine-tuning, Latent Diffusion Models, Generative Models, Neural Networks, AdamW Optimizer, Image Generation, Architecture styles, Machine Learning, Training Data, Validation Data, Civil Engineering.

Introduction

The research objective is to evaluate the effectiveness of LoRA for fine-tuning large-parameter neural networks on specialized data. Artificial Neural Networks (ANNs) are rapidly advancing, offering new possibilities in various fields, including structural mechanics. Different types of neural networks have the potential to significantly enhance the efficiency and accuracy of calculations in structural mechanics. Multilayer perceptions (MLPs) are used to approximate nonlinear functions that describe the behavior of materials and structures. Convolutional Neural Networks (CNNs) enable the analysis of structural images to detect defects and predict material behavior. Recurrent Neural Networks (RNNs) can model the dynamic behavior of structures under loads, including random and impact loads. Deep Neural Networks (DNNs) can solve complex nonlinear problems, taking into account physically nonlinear material properties, geometrically nonlinear deformation, and other complex factors. Latent Diffusion Models (LDMs) have shown significant success in generating high-quality images and can be effectively used for prototyping, creating new designs, visualizing projects, and generating ideas in architecture and construction [1–4]. Large Language Models (LLMs) can normalize data representation, generate hypotheses, assist in programming and design, create reports, improve communication, and be used in solving specialized equations in structural mechanics. Overall, neural networks offer a powerful toolkit for enhancing the accuracy and efficiency of calculations in structural mechanics, allowing for the consideration of complex factors and the processing of large datasets [5–8].A recent trend in neural network research is the creation of complex ecosystems involving different types of ANNs to solve complex problems. Of course, in structural mechanics, material strength, and other construction disciplines, specialized models trained on domain-specific data, designed and parameterized for construction and architecture tasks, are primarily needed. Such tools are being created and will continue to be created. However, training neural network models with

a large number of parameters presents a challenge. Building modern LLMs and LDMs from scratch requires massive datasets, significant computational and time costs, and can cost millions of dollars. However, there is an opportunity to fine-tune existing models, adapting them to specific tasks and data. One convenient approach for fine-tuning is the Low-Rank Adaptation (LoRA) method, which allows for adjusting the model by adding a small number of parameters. The LoRA method was initially proposed for LLMs [21], but this article will explore how this method can be used to fine-tune LDMs.

Fine-tuning of Neural Networks

Fine-tuning neural networks is an effective strategy for adapting models to specific tasks and data. This approach is particularly relevant in architecture and construction, where generating images of buildings requires consideration of architectural styles, spatial planning solutions, structural features, and other specific requirements [9–14]. Unlike training from scratch, which involves adjusting all model parameters, fine-tuning focuses on refining existing knowledge. This significantly reduces the time and computational resources needed for model adaptation. Fine-tuning can be compared to specialization, where the base model acquires additional knowledge and skills to address a specific task. In the context of generating images of buildings, fine-tuning allows for adapting the model to various architectural styles, such as Gothic, Baroque, Modern, etc. A model initially trained on a large dataset of diverse images can be fine-tuned on a smaller dataset of images featuring buildings in a particular style. This enables the model to generate new images that adhere to the given style and incorporate its characteristic features. A key advantage of fine-tuning is the ability to reuse already trained network layers. This allows the model to quickly adapt to new data, as it does not need to relearn basic features. Instead, the model focuses on adjusting parameters responsible for the specific characteristics of the task. LoRA, as a fine-tuning method, offers even greater efficiency by focusing on adding a small number of parameters to the model instead of adjusting all weights. This significantly reduces computational costs and training time. LoRA democratizes the use of LDMs in architecture, allowing for customizing models to specific needs without requiring powerful hardware or large datasets.

The main advantages of fine-tuning are as follows:

- Flexibility: Allows for adapting the model to specific tasks.
- Efficiency: Requires fewer computational resources compared to training a model from scratch.
- Reduced training time: Significantly shortens the time needed to adapt the model to a new task.

LoRA: Low Rank Adaptation

LoRA is a fine-tuning method for large models that adapts the model to new data by adding a small number of parameters. The core idea of LoRA is that the weights of the model layers can be represented through low-rank matrices. Consider a linear layer of a neural network without an activation function, which performs the mapping $x \to y = Wx$, where W is the weight matrix. We will modify the operation of this layer by fine-tuning the model, adjusting the weights by ΔW so that the new output is:

$$x \to y' = W'x = (W + \Delta W)x = y + \Delta W x. \quad (1)$$

The term $\Delta W x$ can be interpreted as the result of another, separate, fully connected layer. In other words, we can fix the weights of the matrix W and train ΔW as a model that predicts the difference between the output of the original model and the fine-tuned one. The matrices W and ΔW must be of the same size, for example, 128 by 128, but the rank of ΔW, i.e., the number of linearly independent rows or columns, can be less than 128, for example, 4 or 8. We can represent the matrix ΔW as a product of matrices A and B, significantly reducing the number of parameters for training if, for example, the size of A is 128 by 2, and the size of B is 2 by 128. That is, instead of $128^2 = 16384$, we will have $128 \cdot 2 + 2 \cdot 128 = 512$. In the general case, we will have to train $(nr + rn)/n^2 = 2r/n$ fewer parameters, but we limit the potential of fine-tuning by the low rank of ΔW. Note that during fine-tuning, we need to store the weights of the original model W i ΔW in the computer's memory, but calculate gradients only for the "small" matrices A and B.

More formally, for each layer of the model with weights $W \in \mathbf{R}^{d \times k}$, LoRA introduces two matrices $W \in \mathbf{R}^{d \times r}$ and $B \in \mathbf{R}^{r \times k}$, where $r << \min(d,k)$. The output of the layer is modified as follows:

$$h = Wx + BAx, \tag{2}$$

where x is the input of the layer, and h is the output of the layer. During fine-tuning with LoRA, the original weights W remain unchanged, and the matrices A and B are trained on new data. This allows for a significant reduction in the number of parameters that need to be trained, which speeds up the fine-tuning process and reduces computational costs.

The advantages of the LoRA method are:

- Significant reduction in the number of parameters that need to be trained, which speeds up the fine-tuning process and reduces computational costs.
- Applicability to different types of model layers, such as fully connected layers and convolutional layers.
- Easy implementation and integration into existing deep learning frameworks.

Before discussing the process of fine-tuning a specific large model, namely Stable Diffusion 1.5, we will describe the structure of the diffusion model and the mathematical principles underlying it.

Mathematical Foundation of Diffusion Models

Diffusion models are based on the idea of gradually adding noise to an image until it turns into pure noise. The model is then trained on the reverse process – removing noise from random noise to restore the original image [15–17].

Forward Diffusion Process. The forward diffusion process can be described as a Markov chain, where a small amount of Gaussian noise is added to the image at each step. Formally, this process can be described as follows:

$$q(x_t | x_{t-1}) = N(x_t; \sqrt{1-\beta_t} x_{t-1}, \beta_t I), \tag{3}$$

where x_0 is the original image; $x_1, \ldots, x_T$ is the sequence of noisy images; $\beta_1, \ldots, \beta_T$ are hyper parameters that determine the amount of noise added at each step; these values usually increase linearly or according to a given schedule from a small value (e.g., 0.0001) to a larger one (e.g., 0.02); $N(x; \mu, \sigma)$ is the Gaussian distribution with mean μ and variance σ; I is the identity matrix.

This process can be interpreted as a gradual "forgetting" of information about the original image. With each step, the image becomes increasingly noisy until it turns into pure Gaussian noise. Using reparameterization, a formula for the forward process can be derived that depends only on the input image x_0:

$$x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t} \phi, \tag{4}$$

where $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$; $\alpha_t = 1 - \beta_t$; ϕ is Gaussian noise.

This formula allows for significantly speeding up the forward process, since x_t can be directly sampled at any time step.

Reverse Diffusion Process. The reverse diffusion process is modeled using a neural network that is trained to remove noise from the image. This process can be described as follows:

$$p_\theta(x_{t-1} | x_t) = N(x_{t-1}; \mu_\theta(x_t, t), \Sigma_\theta(x_t, t)), \tag{5}$$

where θ represents the parameters of the neural network; $\mu_\theta(x_t, t)$ is the predicted mean of the distribution, which is calculated by the neural network based on the noisy image x_t and the time step t; $\Sigma_\theta(x_t, t)$ is the predicted variance of the distribution. In many implementations of diffusion models, Σ_θ is considered a constant or a fixed function of β_t.

The neural network is trained to predict the parameters of the distribution $p_\theta(x_{t-1} | x_t)$, which allows it to gradually remove noise from the image and restore the original image.

To implement the reverse process, it is necessary to define the conditional distribution $p(x_{t-1} | x_t)$. Using Bayes' theorem and the Markov property of the diffusion process, it can be shown that:

$$p(x_{t-1} | x_t) = \int p(x_{t-1} | x_t, x_0) p(x_0 | x_t) dx_0, \tag{6}$$

where $p(x_{t-1} \mid x_t, x_0)$ is the distribution of x_{t-1} given x_t and x_0, and $p(x_0 \mid x_t)$ is the posterior distribution of x_0 given x_t [18].

Since the forward diffusion process is a Markov chain with Gaussian transitions, the distribution $p(x_{t-1} \mid x_t, x_0)$ is also Gaussian and can be computed analytically. However, the posterior distribution $p(x_0 \mid x_t)$ is difficult to compute. Therefore, in diffusion models, this distribution is approximated using a neural network.

Given that x_0, x_t and x_{t-1} are related as (4), we can express x_0 in terms of x_t and ϕ:

$$x_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\phi}{\sqrt{\bar{\alpha}_t}}. \tag{7}$$

Substituting this value into the formula for $q(x_{t-1} \mid x_{t,}, x_0)$, we get:

$$q(x_{t-1} \mid x_t, x_0) = N(x_{t-1}; \tilde{\mu}_t(x_t, x_0), \tilde{\beta}_t I), \tag{8}$$

where

$$\tilde{\mu}_t(x_t, x_0) = \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})x_t + \sqrt{\alpha_{t-1}}\beta_t x_0}{1-\bar{\alpha}_t}, \tag{9}$$

$$\bar{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t. \tag{10}$$

Training the Diffusion Model is based on maximizing the negative log-like lihood of the training data. After a series of calculations, the Evidence Lower Bound (ELBO) can be written as:

$$\log p(x) \geq \mathbf{E}_{q(x_1|x_0)}\left[\log p_\theta(x_0 \mid x_1)\right] - \mathbf{D}_{KL}\left(q(x_T \mid x_0) \| p(x_T)\right) -$$
$$- \sum_{t=2}^{T} \mathbf{E}_{q(x_t|x_0)}\left[\mathbf{D}_{KL}\left(q(x_{t-1} \mid x_t, x_0) \| p_\theta(x_{t-1} \mid x_t)\right)\right] = L_0 - L_T - \sum_{t=2}^{T} L_{t-1}, \tag{11}$$

where $\mathbf{E}$ is the mathematical expectation over the distribution; $\mathbf{D}_{KL}(P \| Q)$ is the Kullback-Leibler divergence, a measure of how much a model probability distribution Q is different from a true probability distribution P; L_0 is the reconstruction term, similar to the ELBO of a variational autoencoder; L_T shows how close x_T is to the standard Gaussian distribution. This term has no learnable parameters, so it is ignored during training; $\sum_{t=2}^{T} L_{t-1}$ formulates the difference between the desired denoising steps $p_\theta(x_{t-1} \mid x_t)$ and the approximated ones $q(x_{t-1} \mid x_t, x_0)$.

From the expression for ELBO, it is clear that maximizing the likelihood is reduced to learning the denoising steps L_t.

Loss Function. To train the diffusion model, the Variational Lower Bound (VLB) on the negative log-likelihood of the data is used. The VLB can be decomposed into several components, one of which is responsible for restoring the image from noise. This component has the form:

$$L_t = \mathbf{E}_{q(x_t|x_0)}[\left\|\mu_\theta(x_t, t) - \tilde{\mu}(x_t, x_0)\right\|^2], \tag{12}$$

where $\tilde{\mu}(x_t, x_0)$ is the optimal mean of the distribution $p(x_{t-1} \mid x_t)$, which can be calculated analytically, knowing the parameters of the forward diffusion process.

By minimizing this loss function, we train the neural network to predict the optimal mean of the distribution, which allows it to effectively remove noise from the image.

In many implementations of diffusion models, the loss function L_t is parameterized as follows:

$$L_t = \mathbf{E}_{x_0, t, \phi}\left[\frac{\beta_t^2}{2\alpha_t(1-\bar{\alpha}_t)\left\|\Sigma_\theta\right\|_2^2}\left\|\phi_t - \phi_\theta\left(\sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\phi, t\right)\right\|^2\right], \tag{13}$$

where ϕ_t is the noise that was added to the image at step t of the forward diffusion process; ϕ_θ is the predicted noise, which is calculated by the neural network based on the noisy image x_t and the time step t.

This parameterization allows for simplifying the calculation of the loss function and improving the stability of training. Additional "penalties" can be added to the standard loss function for diffusion models described above. For example, Weight Decay, a penalty proportional to the square of the magnitude of the weights, is discussed below. This forces the model to prefer smaller weight values, leading to a simpler and more generalized model and preventing overfitting.

Latent Diffusion Models (LDMs) differ from standard diffusion models in that they operate in the latent space, which is accessed through a Variational Autoencoder (VAE). The VAE consists of two parts: an encoder, which transforms the image into a latent representation vector, and a decoder, which restores the image from the latent representation vector. The use of VAE allows LDMs to work with lower data dimensionality, which reduces computational costs and improves generation quality. Instead of applying the diffusion process to high-resolution images, LDMs apply it to compact latent representation vectors. This significantly reduces the computational complexity of the model and improves its efficiency.

Architecture of Latent Diffusion Models

LDMs typically use the U-Net type [19] as the main architecture of the neural network for the reverse diffusion process. U-Net is a convolutional neural network that consists of two parts: an encoder, which reduces the dimensionality of the image by sequentially applying convolutions and pooling operations, and a decoder, which restores the image to its original size by sequentially applying convolutions and upsampling operations. U-Net also uses skip connections, which connect layers of the encoder and decoder with the same resolution. Skip connections allow for transferring information from the encoder to the decoder, which improves the quality of image restoration. In LDMs, U-Net takes a noisy latent representation vector and a time step as input, and outputs the predicted mean of the distribution $p_\theta\left(x_{t-1} \mid x_t\right)$.

AdamW Optimization

Instead of regular stochastic gradient descent (SGD) during backpropagation for training, we will use AdamW (Adam with Weight Decay) optimization, which is an improvement over another adaptive optimization method, Adam (Adaptive Moment Estimation).

SGD is the basic backpropagation method, where parameter updates occur in the direction opposite to the gradient of the loss function at a constant learning rate for all parameters. This method can be slow and get "stuck" in local minima or saddle points.

Adam, compared to SGD, includes the following improvements:

1. **Momentum:** Introduces an exponential moving average (EMA) of gradients, which smooths updates and helps to pass through noise and small local minima.
2. **Adaptive learning rate:** Calculates the EMA of the squares of gradients. This allows for different learning rates for each parameter. Parameters with large gradients receive a smaller learning rate, and parameters with small gradients receive a larger one.
3. **L2-regularization:** Adds a penalty proportional to the square of the weights to the loss function. This "pushes" the absolute values of the weights towards zero, preventing overfitting.

AdamW retains all the advantages of Adam (momentum and adaptive learning rate). The main difference is the decoupling of Weight Decay. AdamW implements L2-regularization (weight decay) not by adding a penalty to the loss function, but by adding it directly to the parameter update step. This ensures that all parameters receive the same penalty for their magnitude, regardless of their gradients. This contributes to the better generalization ability of the model, reducing the error on new data. The penalty for the magnitude of the weights is applied consistently, regardless of the history of gradients. In addition, decoupling weight decay in AdamW makes it easier to tune hyperparameters. The link between the optimal learning rate and the optimal L2-regularization coefficient becomes weaker. This makes AdamW a more convenient and often more effective choice than classic Adam with L2-regularization.

In classic Adam, L2-regularization is added directly to the gradient. That is, we penalize large weights when calculating the gradient

$$g_t = \nabla f\left(\theta_t\right) + w_t \theta_t, \tag{14}$$

ISSN 2410-2547
Опір матеріалів і теорія споруд/Strength of Materials and Theory of Structures. 2025. № 114

where g_t is the modified gradient at time step t. This is what will be used to update the parameters; $\nabla f(\theta_t)$ is the gradient of the loss function f with respect to the model parameters θ at time step t, i.e., the direction in which the loss function increases most rapidly (we want to move against this direction to minimize the loss); θ_t are the parameters (weights) of the model at time step t; w_t is the weight decay rate at time step t (this is a small number, for example 0.01, which controls the strength of L2 regularization); $w_t\theta_t$ is the L2 regularization term (we add a penalty to the gradient, proportional to the magnitude of the weights, forcing the weights to be smaller in absolute value, which helps prevent overfitting).

In AdamW, weight decay ($w_{t,i}\theta_{t,i}$) is added directly to the parameter update, not to the gradient. We first calculate the parameter update using the usual Adam formula, and then apply weight decay

$$\theta_{t+1,i}=\theta_{t,i}-\eta\left(\frac{1}{\sqrt{\hat{v}_t+\varepsilon}}\hat{m}_t+w_{t,i}\theta_{t,i}\right),\forall t, \tag{15}$$

where $\theta_{t+1,i}$ is the value of the i-th parameter at the next time step $t + 1$; $\theta_{t,i}$ is the value of the i-th parameter at the current time step t; η is the learning rate; $\hat{v}_t$ is the bias-corrected second raw moment estimate (essentially, the adaptive learning rate for each parameter); ε is a constant for numerical stability (prevents division by zero, for example 0.0000001); $\hat{m}_t$ is the bias-corrected first moment estimate; $w_{t,i}$ is the weight decay coefficient for the i-th parameter at time step t; $\frac{1}{\sqrt{\hat{v}_t+\varepsilon}}\hat{m}_t$ is the standard part of the Adam formula; $w_{t,i}\theta_{t,i}$ is the decoupled weight decay term.

The biased first and second moment estimates of the gradient are expressed here in terms of the unbiased estimates as follows:

$$m_t=\beta_1 m_{t-1}+(1-\beta_1)g_t, \tag{16}$$

$$\hat{m}_t=\frac{m_t}{(1-\beta_1^t)}, \tag{17}$$

$$v_t=\beta_2 v_{t-1}+(1-\beta_2)g_t^2, \tag{18}$$

$$\hat{v}_t=\frac{v_t}{(1-\beta_2^t)}, \tag{19}$$

where m_t is the biased first moment estimate (exponentially smoothed average of the gradients) at time step t; v_t is the biased second raw moment estimate (exponentially smoothed uncentered variance of the gradients) at time step t.

The advantages of AdamW (in many cases) are:

- Simplifying the training process by decoupling the influence of the learning rate and weight decay, which eliminates the drawback of Adam, where L2-regularization is "mixed" with the adaptive learning rate, leading to inefficient search for optimal parameters.
- Better generalization performance of the model, i.e., the model performs better on new data.

Setting the Test Task and Choosing Software

We will fine-tune the Stable Diffusion 1.5 latent diffusion model, which is capable of generating high-quality images, to generate buildings in several architectural styles. To implement the LoRA method and the AdamW optimizer discussed above, we will use the open-source tool One Trainer [20], which has a user-friendly interface for setting hyperparameters, choosing optimizers and loss functions, monitoring the training process, and supports various types of fine-tuning, including LoRA.

Training

To fine-tune the Stable Diffusion 1.5 model using LoRA in One Trainer, five sets of images of buildings in the styles of Bauhaus, Deconstructivist Architecture, Parametric Architecture, Renaissance, and Neoclassical Architecture were prepared for training the corresponding five LoRA

models. Each set contained high-quality images of 512x512 or 768x512 pixels and consisted of 64 images without noise and artifacts. All images were annotated (accompanied by a text description) using the Blip2 model (Bootstrapping Language-Image Pre-training for Unified Vision-Language, v2). In each of the 5 sets, a validation part of 8 images was allocated, and the remaining 56 images formed the training dataset.

The following hyperparameters were set for training: Batchsize: 4; Learningrate: 0.0003; Optimizer: AdamW with weight coefficients from formulas 15-19: $w_{t,i} = 0.01$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 0.00000001$; Loss function: standard loss function for diffusion models (taking into account AdamW); LoRA rank (see the section "LoRA: Low Rank Adaptation" above): 16.

In addition to the above features of the applied AdamW optimization, the training process followed the usual scheme:

1. **Forward propagation:** Images from the dataset are fed to the input of the model. The model generates images that will be compared with the original images.
2. **Loss calculation:** The difference between the generated and original images is calculated.
3. **Backpropagation:** The losses are used to update the weights of the LoRA matrices.

The epoch, i.e., steps 1-3 performed for all images of the training dataset, thus consisted of 56/4=14 steps. To ensure a more stable start to the training process, the learning rate was linearly increased from 0 to 0.0003 over the first 100 steps. After each epoch, a validation step took place, i.e., calculating the loss function on data unknown to the model during training. The process was monitored using Tensor Flow. The values of the loss function for the training and validation sets were passed to Tensor Flow after each step, and test image generation with fixed prompt, seed, and other generation parameters was performed after every 5 epochs.

Figures 2a and 2b illustrate the smoothed loss function values for the training and validation sets of the Bauhaus style over 200 epochs. Figure 2c displays test generations for the simple prompt "building" with fixed generation parameters. The oscillatory pattern in Figure 2a indicates an aggressively high learn in grate, leading to faster but potentially unstable training. The progression of the training process is evident from observing the test generations, which visually mimic the characteristics of the dataset images.

The validation loss function graph in Figure 2b is significantly more informative. Region *O* can be interpreted as the classic overfitting zone, where the model starts to memorize noise instead of general patterns. In this case, it implies that the LoRA model is learning to reproduce specific elements of individual images rather than the overall style inherent in the training dataset. This explains the increasing discrepancy when comparing generations from this stage with the validation dataset, which shares general characteristics with the training set but lacks the specific features memorized from the training images. The local minima in region *G* are suitable candidates for stopping the training process. We chose point *p* to terminate training. Similarly, loss function graphs for the other four LoRA models, with the priority region *G* and the chosen training stop point *p* marked, are shown in Figures 2d-2g. We attribute the stability of the training process with such a high learning rate to the adaptive nature of the AdamW optimizer.

The resulting LoRA models contain fewer than 20 million parameters and have a size of 78,489,960 bytes. This is significantly smaller than the size of the main SD 1.5 model with approximately 860 million parameters and a size of 4,265,146,304 bytes.

Figure 3 presents examples of images generated using the fine-tuned LoRA models with the prompt "Building of the city theater, summer atmosphere" and Control Net with Lineart and Depth models for the image of the Kyiv Theater on Podil. A soft mask image was used to define the generation zone based on the original image. In other words, the task was to generate the appearance of the Theater on Podil building in the styles of Bauhaus, Deconstructivist Architecture, Parametric Architecture, Renaissance, and Neoclassical Architecture, while preserving the overall compositional and spatial planning parameters.

Conclusions

Fine-tuning models with a large number of parameters using the Low-Rank Adaptation method is an effective strategy for adapting large models to solve specialized tasks. Fine-tuning the latent diffusion model using the LoRA method can effectively generate images of buildings in specified architectural styles.

ISSN 2410-2547
Опір матеріалів і теорія споруд/Strength of Materials and Theory of Structures. 2025. № 114

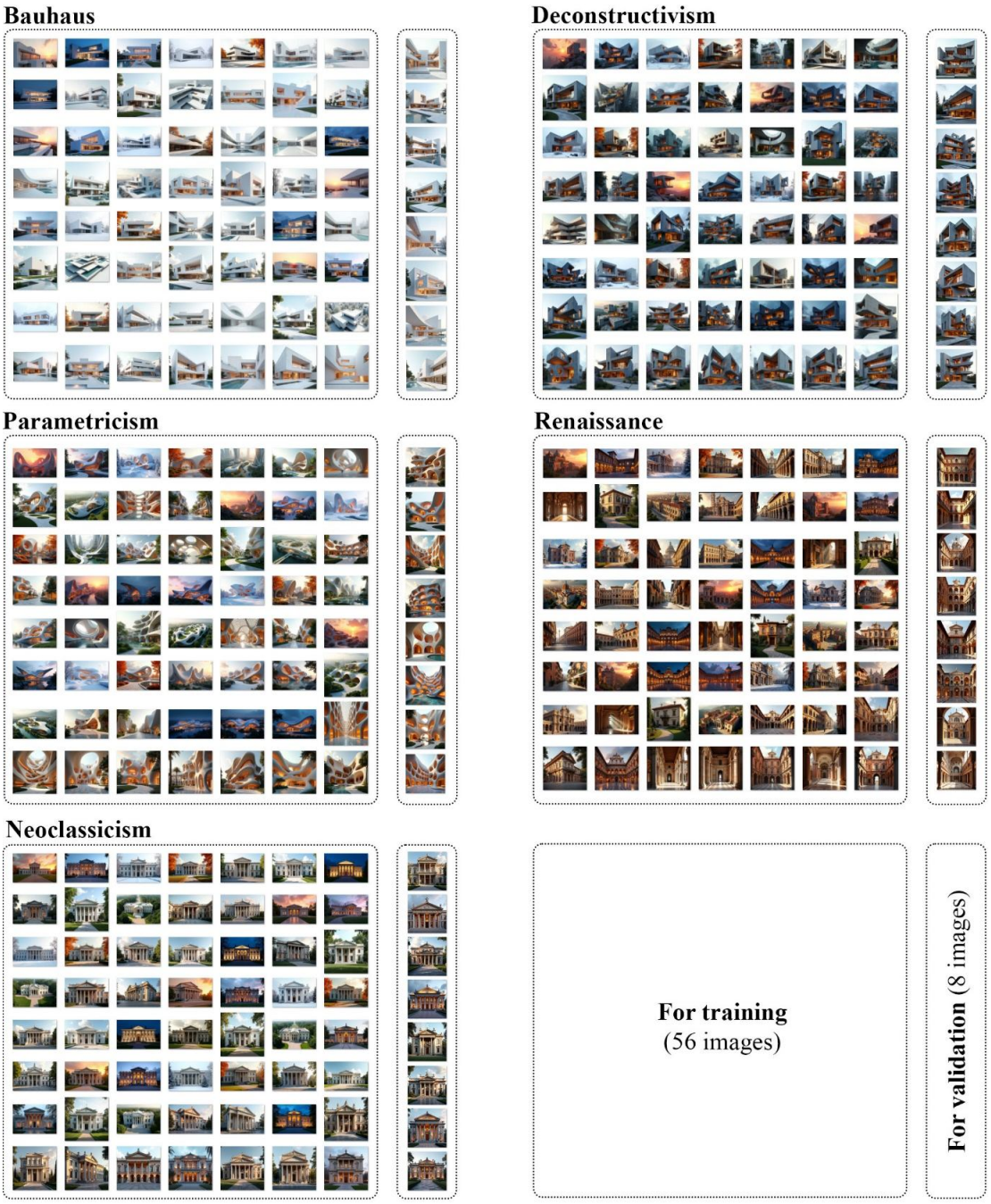

Fig. 1. Datasets of building images in various architectural styles for training with the LoRA method

Analysis of the training process using a validation sample allows for avoiding over fitting of the resulting models, while main taining flexibility in the practical use of LDM + LoRA. LoRA allows for adapting the model to new data by adding a small number of parameters, which significantly reduces computational costs and training time. One Trainer provides convenient tools for fine-tuning LDMs using LoRA. The results of the study showed that fine-tuning Stable Diffusion 1.5 using LoRA allows for generating high-quality images of buildings in selected architectural styles. The model successfully reproduces the characteristic features of the specified styles, which indicates the high potential of using LDMs in architecture and construction.

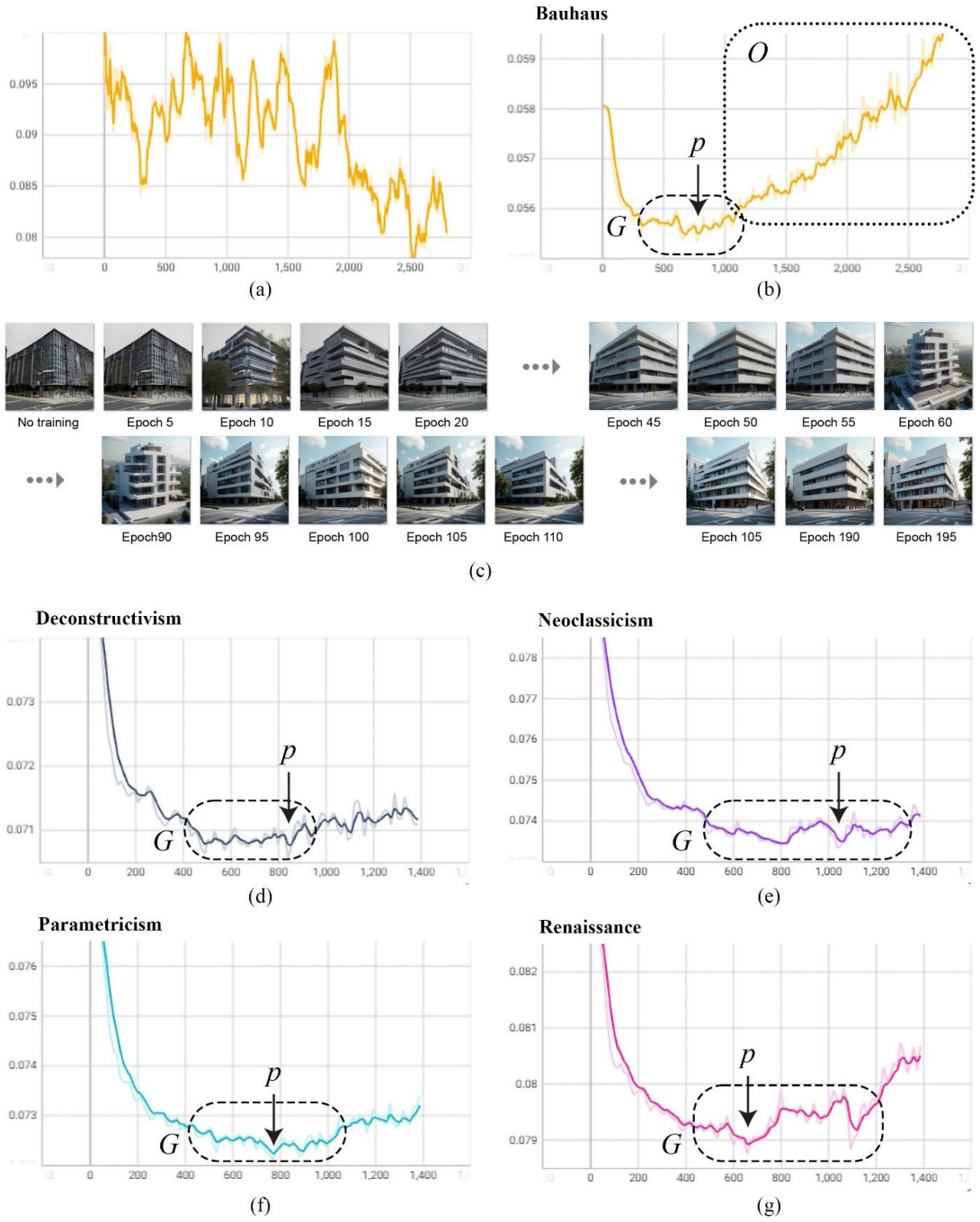

Fig. 2. Features and key indicators of the LoRA training process

Further research plans include:

- Applying LDMs to create specified spatial planning solutions, building and structural systems, and selecting the type of building structures.
- Testing the fine-tuning of other large models using the LoRA method, namely LLMs, for solving specialized problems in structural mechanics.

ISSN 2410-2547
Опір матеріалів і теорія споруд/Strength of Materials and Theory of Structures. 2025. № 114

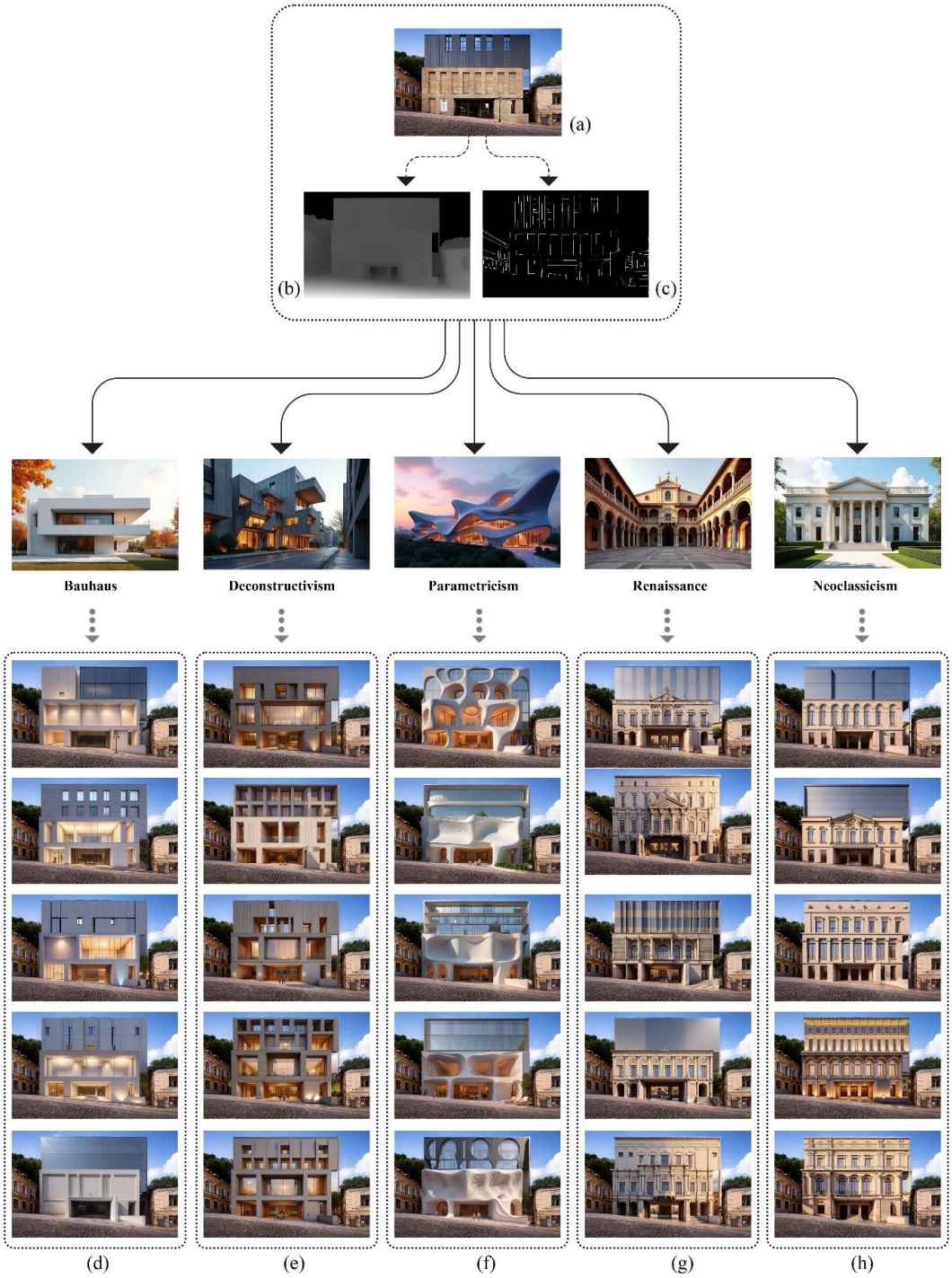

Fig. 3. Generations of the Kyiv Theater on Podil building in various architectural styles using trained LoRA models

- Expanding the application of neural networks for automated analysis of building technical conditions through transfer learning. This will encompass identifying and classifying defects from diverse data sources, including images acquired from sources such as drones, smartphones, and surveillance cameras, as well as data from sensors, ultimately contributing to an integrated system for comprehensive assessment and prediction of building structure conditions.

REFERENCES

1. *Getun, H.V., Ivanchenko, H.M., Skliarov, I.O., Solomin, A.V., &Hetun, S.Y.* Zastosuvannya neyromerezh v arkhitekturi budivel' I optymizatsiya dlya ts'oho modelei prykhovanoi dyfuzii (Application of neural networks in building architecture and optimization of latent diffusion models for this purpose). Kyiv: Current problems of architecture and urban planning, 2025, Vol. 71, P. 494-509, DOI:10/32347/2077-3455.2025.71.494-509 (Ukraine)
2. *Wenjie Liao, Xinzheng Lu, Yifan Fei, Yi Gu, Yuli Huang.* Generative AI design for building structures. Automation in Construction. 2024, Vol. 157, 105187. ISSN 0926-5805. DOI: 10.1016/j.autcon.2023.105187. Netherlands**,** Amsterdam: Elsevier B.V. (Netherlands)
3. *Hao Leng, Yuqing Gao, Ying Zhou.*ArchiDiffusion: A novel diffusion model connecting architectural layout generation from sketches to Shear Wall Design. Journal of Building Engineering. 2024, Vol.98,111373.ISSN 2352-7102. DOI: 10.1016/j.jobe.2024.111373.Netherlands: Elsevier B.V. (Netherlands)
4. *Zhuang Tan, Sizhong Qin, Kongguo Hu, Wenjie Liao, Yuan Gao, Xinzheng Lu*. Intelligent generation and optimization method for the retrofit design of RC frame structures using buckling-restrained braces. Earthquake Engineering and Structural Dynamics. 2024, 14 November. DOI: 10.1002 / eqe. 4268. United Kingdom: John Wiley and Sons Ltd. (United Kingdom)
5. *Botvinovska S., Getun G., Zolotova A., Korbut I., Nikolaenko T., Parnenko V., Rodin R.* General procedure for determining the geometric parameters of tools in the technological systems involving machining by cutting. Eastern-European Journal of Enterprise Technologies. Vol. 1 No. 1 (109) (2021): Engineering technological systems. Published: 2021-02-19. year 6-12. UDC 621.9 DOI: 0.15587/1729-4061.2021.224897 (Ukraine)
6. *Ivanchenko G.M., Getun G.V., Bezklubenko I.S., Solomin A.V., Getun S.Y.* Mathematical model of the stress-strain state of multilayered structures with different elastic properties // Strength of Materials and Theory of Structures: Scientific-&-Technical collected articles – Kyiv: KNUBA, 2024. – Issue 113. – P. 131-138. http://opir.knuba.edu.ua/ http:/omtc.knuba.edu.ua/ DOI: 10.32347/2410-2547.2024.113.131-138 (Ukraine)
7. *Getun G.V., Butsenko Y.P., Labzhynsky V.A., Balina O.I., Bezklubenko I.S., SolominA.V.* Prognozuvannya sytuatsiy ta optymizatsiya pryinyattya rishen' na osnovi skinchennykh lantsihiv Markova v raionakh z promislovym zabrudnennyam (Situation forecasting and decision-making optimizations based on using markov finite chains for areas with industrial polutions). // Strength of Materials and Theory of Structures: Scientific-&-Technical collected articles – Kyiv: KNUBA, K: 2020. Issue 104. - P. 164-174. ISSN 2410-2547. DOI: 10.32347/2410-2547.2020.104.164-174 (Ukraine)
8. *Ying Zhou, Hao Leng, Shiqiao Meng, Hao Wu, Zheng Zhang*. StructDiffusion: End-to-end intelligent shear wall structure layout generation and analysis using diffusion model. Engineering Structures, Volume 309, 2024, 118068. ISSN 0141-0296. DOI: 10.1016/j.engstruct.2024.118068. United Kingdom: Elsevier B.V. (United Kingdom)
9. *R. Rombach, A. Blattmann, D. Lorenz, P. Esserand B. Ommer.* High-Resolution Image Synthesis with Latent Diffusion Models. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR*)*, NewOrleans, LA, USA, 2022, pp. 10674-10685, DOI: 10.1109/CVPR52688.2022.01042 (UnitedStatesofAmerica).
10. *Ho, Jonathan, Ajay Jain, and Pieter Abbeel.* Denoising diffusion probabilistic models. Advances in neural information processing systems, 2020, 33: 6840-6851.. https://doi.org/10.48550/arXiv.2006.11239
11. *Song, Yang, and Stefano Ermon*. Improved techniques for training score-based generative models. Advances in neural information processing systems, 2020, 33: 12438-12448. https://doi.org/10.48550/arXiv.2006.09011
12. *Weng, Lilian*. What are diffusion models? Lil'Log. 2021, Jul.https://lilianweng.github.io/posts/2021-07-11-diffusion-models/.
13. *Ronneberger, Olaf, Philipp Fischer, and Thomas Brox.* U-net: Convolutional networks for biomedical image segmentation. Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5–9, 2015, proceedings, part III 18. Springer International Publishing, 2015. https://doi.org/10.48550/arXiv.1505.04597.
14. One Trainer. https://github.com/Nerogar/OneTrainer
15. *Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... & Chen, W.*LoRA: Low-rank adaptation of Large Language Models. arXiv preprint arXiv:2106.09685, 2021.https://doi.org/10.48550/arXiv.2106.09685.

Стаття надійшла 13.02.2025

Іванченко Г.М., Гетун Г.В., Скляров І.О., СоломінА.В., Гетун С.Ю.

ЗАСТОСУВАННЯ МЕТОДУ НИЗЬКОРАНГОВОЇ АДАПТАЦІЇ НА ПРИКЛАДІ ДОНАВЧАННЯ МОДЕЛІ ПРИХОВАНОЇ ДИФУЗІЇ

Стаття досліджує метод низькорангової адаптації (LoRA), швидку методику тонкого налаштування нейронних мереж з великою кількістю параметрів, та її потенційне застосування в різних галузях, з акцентом на архітектуру, будівництво та будівельну механіку. Дослідження застосовує LoRA для тонкого налаштування моделі прихованої дифузії (LDM) для генерації зображень будівель у різних архітектурних стилях, слугуючи ілюстративним прикладом ефективності LoRA для адаптації великих моделей до спеціалізованих завдань.

Нейронні мережі з великою кількістю параметрів, такі як моделі прихованої дифузії (LDM) та великі мовні моделі (LLM), продемонстрували значний потенціал у різних галузях, але їх навчання з нуля є обчислювально дорогим та трудомістким. Тонке налаштування пропонує більш ефективний підхід шляхом адаптації попередньо навчених моделей до конкретних завдань та даних. LoRA ще більше підвищує ефективність, додаючи невелику кількість параметрів до моделі, а не налаштовуючи всі ваги. LoRA використовує низькорангові матричні представлення для зменшення кількості параметрів, що навчаються, під час тонкого налаштування. Вводячи менші матриці для кожного шару та навчаючи їх на нових даних, LoRA значно прискорює процес тонкого налаштування та зменшує обчислювальні витрати. Дослідження демонструє застосування LoRA для тонкого налаштування LDM StableDiffusion 1.5 для генерації зображень будівель у різних архітектурних стилях за допомогою інструменту OneTrainer.

Результати показують, що тонке налаштування StableDiffusion 1.5 за допомогою LoRA ефективно генерує високоякісні зображення будівель у заданих архітектурних стилях, підкреслюючи потенціал LoRA для адаптації великих моделей до спеціалізованих завдань. Наголошується на використанні набору даних для валідації для запобігання перенавчанню та визначення оптимальної точки зупинки навчання, забезпечуючи узагальненість моделі.

Це дослідження робить внесок у ширше дослідження застосовності LoRA для тонкого налаштування великих нейронних мереж у різних областях. Хоча дослідження зосереджено на LDM для архітектурних застосувань, основні принципи та

продемонстрована ефективність LoRA поширюються на інші типи великих моделей, такі як LLM, для вирішення спеціалізованих завдань у різних галузях.

Ключові слова: метод низькорангової адаптації, тренування нейромереж, моделі прихованої дифузії, генеративні моделі, нейромережі, оптимізатор AdamW, генерація зображень, архітектурні стилі, машинне навчання, датасет, валідаційний датасет.

Ivanchenko H.M., Getun G.V., Skliarov I.O., Solomin A.V., Getun S.Y.

APPLICATION OF THE LOW-RANK ADAPTATION METHOD ON THE EXAMPLE OF FINE-TUNING A LATENT DIFFUSION MODEL

This article explores the Low-Rank Adaptation (LoRA) method, a fast fine-tuning technique for large-parameter neural networks, and its potential application in various fields, with a focus on architecture, construction, and structural mechanics. The study applies LoRA to fine-tune a Latent Diffusion Model (LDM) for generating images of buildings in various architectural styles, serving as an illustrative example of LoRA's effectiveness for adapting large models to specialized tasks.

Large-scale neural networks, such as Latent Diffusion Models (LDMs) and Large Language Models (LLMs), have shown significant potential in various fields, but their training from scratch is computationally expensive and time-consuming. Fine-tuning offers a more efficient approach by adapting pre-trained models to specific tasks and data. LoRA further enhances efficiency by adding a small number of parameters to the model instead of adjusting all weights. LoRA uses low-rank matrix representations to reduce the number of trainable parameters during fine-tuning. By introducing smaller matrices for each layer and training them on new data, LoRA significantly speeds up the fine-tuning process and reduces computational costs.

The study demonstrates the application of LoRA for fine-tuning the LDM Stable Diffusion 1.5 to generate images of buildings in various architectural styles using the OneTrainer tool. The results show that fine-tuning Stable Diffusion 1.5 using LoRA effectively generates high-quality images of buildings in specified architectural styles, highlighting LoRA's potential for adapting large models to specialized tasks. The use of a validation dataset is emphasized for preventing overfitting and determining the optimal stopping point for training, ensuring the model's generalizability.

This research contributes to the broader exploration of LoRA's applicability for fine-tuning large neural networks in various domains. While the study focuses on LDMs for architectural applications, the underlying principles and demonstrated effectiveness of LoRA extend to other types of large models, such as LLMs, for addressing specialized tasks in different fields.

Keywords: Low-Rank Adaptation, Fine-tuning, Latent Diffusion Models, Generative Models, Neural Networks, AdamW Optimizer, Image Generation, Architecture styles, Machine Learning, Training Data, Validation Data.

УДК 539.3
Іванченко Г.М., Гетун Г.В., Скляров І.О., Соломін А.В., Гетун С.Ю. **Застосування методу низькорангової адаптації на прикладі донавчання моделі прихованої дифузії** // Опір матеріалів і теорія споруд: наук.-тех. збірн. – К.: КНУБА. – 2025. – Вип. 114. – С. 299-310.
Досліджується метод швидкого донавчаннянейромереж з великою кількістю параметрів та його застосування в будівництві та архітектурі
Іл. 3. Бібліогр. 15 назв.

UDC 539.3
Ivanchenko H.M., Getun G.V., Skliarov I.O., Solomin A.V., Getun S.Y. **Application of the low-rank adaptation method on the example of fine-tuning a latent diffusion model** // Strength of Materials and Theory of Structures: Scientific-and-technical collected articles. – K.: KNUBA. – 2025. – Issue 114. – P. 299-310.
This paper explores a method for rapid retraining of large-scale neural networks and its applications in construction and architecture.
Figs. 3. Refs. 15.

Автор (науковий ступінь, вчене звання, посада): доктор технічних наук, професор кафедри будівельної механіки Київського національного університету будівництва і архітектури Іванченко Григорій Михайлович
Адреса робоча: 03680 Україна, м. Київ, проспект Повітряних Сил, 31, КНУБА
E-mail: ivanchenko.gm@knuba.edu.ua
ORCID ID: https://orcid.org/0000-0003-1172-2845

Автор (науковий ступінь, вчене звання, посада): кандидат технічних наук, доцент, професор кафедри архітектурних конструкцій Київського національного університету будівництва і архітектури Гетун Галина В'ячеславівна
E-mail: getun.gv@knuba.edu.ua
ORCID ID: http://orcid.org/0000-0002-3317-3456

Автор (науковий ступінь, вчене звання, посада): кандидат технічних наук, доцент, доцент кафедри металевих і дерев'яних конструкцій Київського національного університету будівництва і архітектури Скляров Ігор Олександрович
E-mail: skliarov.io@knuba.edu.ua
ORCID ID: https://orcid.org/0000-0002-6150-5518

Автор (науковий ступінь, вчене звання, посада): кандидат технічних наук, доцент, доцент кафедри біобезпеки і здоров'я людини Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського» Соломін Андрій В'ячеславович
E-mail: a.solomin@kpi.ua
ORCID ID: http://orcid.org/0000-0002-5226-8813

Автор (науковий ступінь, вчене звання, посада): аспірант кафедри будівельної механіки Київського національного університету будівництва і архітектури Гетун Сергій Юрійович
E-mail: hetun_sy-2024@knuba.edu.ua
ORCID ID: https://orcid.org/0009-0001-2269-7035