AFFS: ADAPTIVE FAST FREQUENCY SELECTION ALGORITHM FOR DEEP LEARNING FEATURE EXTRACTION

Anonymous authors
Paper under double-blind review

ABSTRACT

As deep learning(DL) advances, effective feature extraction from big data remains critical for enhancing DL model's performance. This paper proposes a method for feature extraction in the frequency domain, utilizing advantages such as concentrated signal energy and pronounced data features. However, existing frequency component selection algorithms face challenges like difficulty adapting to diverse tasks and achieving only locally optimal results with extended processing times. To address these challenges, we introduce the Adaptive Fast Frequency Selection (AFFS) algorithm, tailored for various subsequent tasks. AFFS incorporates a frequency component selection factor layer, integrating it with the subsequent DL model to select globally optimal frequency component combinations for the DL model. Additionally, we propose a fast selection algorithm to expedite the process, leveraging the experimental observation of rapid convergence of selection factor ranking. Experimental results demonstrate that AFFS achieves superior performance across three datasets and three DL models. By using AFFS to select appropriate frequency components, even though our input data size is only 10% of the original frequency feature, the classification accuracy of the model is improved by about 1%. Furthermore, the early stopping mechanism can shorten the selection process by approximately 80%.

1 INTRODUCTION

The rapid advancement of deep learning (DL) techniques has garnered increasing interest for handling data in various applications, such as sentiment analysis (Ma et al., 2023), speech recognition (Li et al., 2022; Dhanjal & Singh, 2024), object detection (Chan et al., 2023; Gui et al., 2024), and video recognition (Yan et al., 2022). Typically, these studies tackle data handling in two steps: extracting features from the data and subsequently inputting these features into DL models to achieve various objectives.

Feature extraction has become a crucial step in DL-based applications, directly impacting their performance. While traditional methods usually encode data from the original high-dimensional space to low-dimensional spatial-temporal feature embeddings, they often neglect the information in the frequency domain and struggle with issues like noise and outliers.

Recent studies (Kong et al., 2023; Patro et al., 2023; Zhou et al., 2022) advocate transforming data to the frequency domain for feature extraction. Some application examples include: **Image Domain:** JPEG (Hudson et al., 2017) uses the Discrete Cosine Transform (DCT) to compress images by retaining significant frequency components and discarding high-frequency parts. **Speech Signal Domain:** Techniques like the Short-Time Fourier Transform (STFT) (Zhou et al., 2022; Kawamura et al., 2023; Kadiri et al., 2023) enable spectral analysis of speech signals for applications such as coding, noise reduction, and recognition. **Sensor Data Domain:** Frequency-domain preprocessing for sensor data, such as environmental and health monitoring, uses methods like the Fourier Transform to enhance feature extraction and reduce noise (Xu et al., 2023).

These applications demonstrate that utilizing feature extraction in the frequency domain can enhance task performance. The main advantages over direct processing in spatial or temporal domains can be summarized as follows:

Energy Concentration: Frequency domain transformations often concentrate the signal's energy into a few major frequency components, facilitating data compression by retaining these key components.

Enhanced Feature Extraction: Techniques like the Fourier Transform can separate different frequency components, making certain features more prominent.

After frequency transformation, data typically consists of multiple frequency components. Some studies (El Qacimy et al., 2014; Fu & Guimaraes, 2016) have found that DL models exhibit varying sensitivities to different frequency components. Thus, selecting the most effective frequency components for tasks like image classification (Maurício et al., 2023; Bharadiya, 2023), speech recognition (Jeon et al., 2023a;b), and time series prediction (Morid et al., 2023; Dudukcu et al., 2023; Ruan et al., 2023) is crucial.

Consequently, various studies (El Qacimy et al., 2014; Qin et al., 2021; dos Santos et al., 2020; Xu et al., 2020) propose some methods to select the more important subsets of frequency components. These methods aim to

extract important features and reduce redundancy by selecting various sets of frequency components. Among these, most of them focus on the fixed frequency selection methods. For instance, DCT upper left corner (ULC) coefficients (El Qacimy et al., 2014; Fu & Guimaraes, 2016; dos Santos et al., 2020), DCT zigzag coefficients. However, they suffer from the following problems:

Non-Adaptive: They often rely on fixed components that are not optimized for the specific subsequent task. Utilizing identical frequency components across different tasks can lead to suboptimal model performance.

Apart from fixed selection methods, some studies(Qin et al., 2021; Xu et al., 2020) propose learning-based methods to adaptively select frequencies for different subsequent tasks. Although promising, they still have some problems:

Local Optima: They often struggle to determine the optimal combination of frequency components simultaneously. Typically, these methods employ a greedy algorithm that sequentially selects the optimal frequency components (e.g., first selecting the component that yields the best performance for the subsequent tasks, then the next best component, and so forth). However, this approach may result in a locally optimal solution rather than a globally optimal one.

Slow Selection Speed: Their approach involves continuously evaluating the model performance achievable with each selected frequency component, ultimately determining the optimal combination of frequency components through a greedy method. This makes the frequency component selection process computationally expensive. Therefore, designing a fast selection method is crucial to accelerate this process.

1.1 The Contributions of Our Method

To address the aforementioned challenges, we propose the Adaptive Fast Frequency Selection (AFFS) algorithm. This algorithm is designed to adaptively select the most critical frequency components for various subsequent tasks. The main steps involve converting data into frequency components using the Discrete Cosine Transform (DCT), quantifying the importance of each component using selection factors, and selecting components based on their values. The key contributions of this paper are as follows:

1) Task-Specific Frequency Component Selection: We design an easily implementable selection factor layer that requires only minor modifications to existing DL models. Specifically, we add a selection factor layer after frequency conversion and before the subsequent DL model. This layer enables adaptive adjustment based on different subsequent tasks, dynamically selecting the optimal combination of frequency components. Importantly, the selection factors are trained concurrently with the parameters of the subsequent DL model, allowing them to automatically adjust to any subsequent task. Once training is complete, we can select the optimal combination of frequency components at once, thereby avoiding the issue of local optima. Moreover, our FCS module is easy to implement and has plug-and-play functionality for various subsequent tasks due to the minor modifications to the subsequent DL model.

2) Fast Frequency Selection Algorithm: We propose a fast frequency selection algorithm to accelerate the selection of the most important frequency components. From extensive experiments, we discover that the ranking of the most important selection factors is typically determined within the initial few iterations of model training, rather than waiting for the model to converge. Moreover, we propose a metric called SFVA to effectively monitor the convergence of the selection factor ranking, and further propose an early stopping mechanism to accelerate the selection of frequency components, which significantly reduces training time while maintaining model performance.

3) Extensive Experimental Validation: We conduct extensive experiments on three datasets(e.g., CIFAR10, ImageNet, NWPU-RESISW45) and three subsequent DL models(e.g., ResNet18, ResNet50, DenseNet121) to verify the effectiveness of our AFFS algorithm. We have the following observations in experimental results:

Accurate selection. Compared with other frequency component selection methods, our AFFS achieves higher performance across different datasets and models, demonstrating its ability to adaptively select the optimal combination of frequency components for various subsequent tasks. **Fast selection.** Our proposed early stopping mechanism reduces training time by over 80%, enabling fast frequency component selection. **Smaller size but higher performance**. After feature extraction using our algorithm, the size of the resulting feature data is reduced by nearly 90% compared to the original frequency feature data. Moreover, even with the much smaller size of the extracted feature data, the model performance improves by nearly 1% compared to using the original data.

Overall, the AFFS algorithm not only enhances the performance of subsequent DL model by selecting the optimal frequency components but also significantly reduces computational requirements, making it able to achieve efficient and effective feature extraction for various subsequent tasks.

2 Related Work

This section reviews the methods of frequency transformation and the techniques for selecting frequency components.

2.1 THE FREQUENCY TRANSFORMATION METHOD

There are various transformation methods available for frequency-domain data processing. The Fourier Transform (FT)(Bracewell, 1989) is widely used for converting signals from the temporal domain to the frequency domain, allowing for the analysis of different frequency components. However, the FT tends to distribute signal energy across a wide range of frequencies, which can lead to inefficiencies in feature extraction. The Wavelet Transform (WT) (Zhang & Zhang, 2019) offers a multi-resolution analysis, providing both temporal and frequency localization. It is particularly useful for analyzing non-stationary signals and capturing transient features but involves higher computational complexity and implementation challenges.

Different from FT and WT, the Discrete Cosine Transform (DCT) is a widely used frequency domain transformation method in signal processing, particularly for digital images and videos, due to its strong energy concentration characteristics and high computational efficiency (Ahmed et al., 1974; Barbero et al., 1992). So we choose to use the DCT over these methods.

With the rise of deep learning, several studies (Ravì et al., 2016; Gueguen et al., 2018; Rajesh et al., 2019) have integrated DCT into DL-based computer vision frameworks. For example, Gueguen et al. (2018) modify the libjpeg library to generate DCT coefficients, applying these coefficients to train the ResNet-50 model for image classification. This approach reduces image decoding time and achieves significant training acceleration by directly using all DCT coefficients for network training. Building on this, Rajesh et al. (2019) propose a novel CNN model called DCT-CompCNN, which can accept both quantized and non-quantized DCT coefficients as input. Their results show that this method can produce performance similar to traditional CNN methods with faster model training speed.

However, these methods use all frequency components for training and inference, without considering the varying impacts of different frequency components on specific tasks. This results in a large amount of redundant information, leading to unnecessary computational overhead. By evaluating the importance of frequency components and selecting key frequency components, we can reduce input data redundancy and improve the performance of the subsequent DL model.

2.2 FREQUENCY SELECTION

After transforming the original data to the frequency domain using DCT, the data comprises multiple frequency components. To extract efficient feature information, it is necessary to select specific frequency components. We categorize frequency component selection methods into the following two types:

Fixed Selection. This method primarily involves selecting fixed low-frequency components in the DCT coefficient matrix. Fu & Guimaraes (2016) introduce a method combining DCT with truncation to accelerate neural network training. They transform the original input into the DCT domain and, based on the energy compaction property of DCT, truncate the upper-left portion of the DCT coefficient matrix, thereby reducing feature dimensions. dos Santos et al. (2020) extend the ResNet-50 network improved by Gueguen et al. (2018) by incorporating a Frequency Band Selection (FBS) technique to select the most relevant DCT coefficients before inputting them into the network. The principle of FBS is that high-frequency information has a minimal visual impact on images, so only the lowest n frequency components from Y, Cr, and Cb (YCrCb color space) are selected. While effective, this method discards high-frequency information, resulting in a loss of image details and failing to consider the different impacts of Y, Cr, and Cb on the model, which may reduce accuracy. These methods do not account for the impact of other frequency components and are tailored for specific tasks, limiting their adaptability to various subsequent tasks.

Learning-Based Selection. Different from the fixed selection methods, Qin et al. (2021) apply DCT to channel attention and proposed a two-stage frequency selection method. They first evaluate the model performance resulting from different frequency components individually. Then, they select frequency components one by one that lead to the current optimal performance, until a sufficient number of components have been chosen. This method aims to find a local optimum at each step without considering the global performance, which always results in the **local optimal** solutions. Moreover, after selecting each frequency component, it is necessary to reassess the impact of all remaining frequency components on the model performance, which is **high computation cost**.

Additionally, Xu et al. (2020) introduce a frequency channel selection gate module, which assigns two numbers to each frequency channel and decides whether to retain the channel by sampling from the Bernoulli distribution Bern(p). Due to the non-differentiable nature of the Bernoulli sampling process, they employ the Gumbel Softmax reparameterization method to update the gate module weights, allowing gradients to backpropagate through the discrete sampling process. Although effective, the gate module's training process has high computational cost too.

To select the optimal combination of frequency components for different subsequent tasks, we propose the Adaptive Fast Frequency Selection (AFFS) algorithm. This method addresses the loss of high-frequency information in the Fixed Selection approach and can adaptively select the optimal frequency combination for any subsequent task. Additionally, we design an early stopping mechanism to accelerate the selection process of frequency components.

3 PROPOSED METHOD

Leveraging the DCT's exceptional energy concentration, which can condense the energy of a signal or image into a few frequency components, we propose a frequency component selection method called Adaptive Fast Frequency Selection (AFFS). This method dynamically selects frequency components for the subsequent task, achieving efficient data feature extraction.

In this paper, we use the image classification task as an example to demonstrate how AFFS operates. By preprocessing image data using the proposed AFFS, we can make minor modifications to existing neural network models (e.g., image classification DL models), significantly reducing the input data size and improving the performance of the subsequent DL model.

Fig. 1 illustrates the architecture of our approach, which comprises three main modules: Data Frequency Domain Transformation (DFDT), Frequency Component Selection (FCS), and Adaptive Channel Matching (ACM). The DFDT and FCS modules primarily address the initial step of feature extraction and selection in deep learning tasks within the frequency domain. Given that mature and commonly used DL models typically have fixed-sized input requirements, the ACM module is designed to adapt the size of the extracted features to the subsequent DL model.

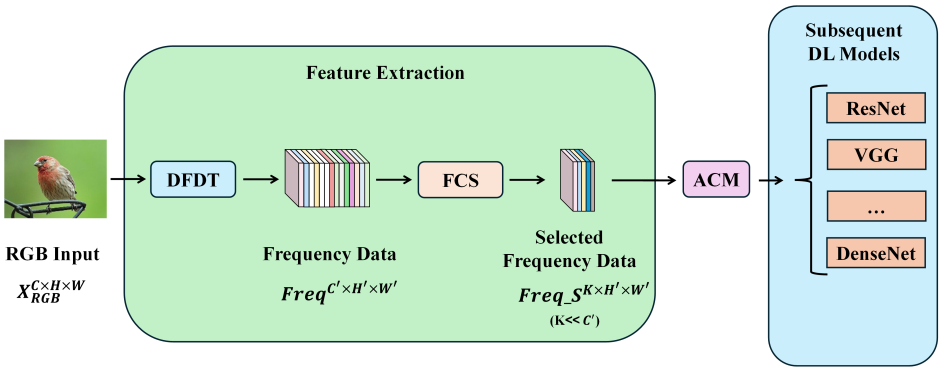


Figure 1: The overall architecture of our approach.

In the DFDT module, we convert original data to the frequency domain using 8×8 DCT and frequency combination. The input to this module is an RGB image $X_{RGB} \in \mathbb{R}^{C\times H\times W}$, where C represents the channels, H represents the height, and W represents the width. The output is the frequency components $Freq \in \mathbb{R}^{C'\times H'\times W'}$, where $C' = 64 \times C$, $H' = H/8$, and $W' = W/8$.

In the FCS module, we introduce a selection factor layer between the DFDT module and the subsequent DL model. This layer includes multiple selection factors, each corresponding to the weight of a frequency component. During training, the DL model and the selection factor layer are trained together. This allows us to adaptively and efficiently select the optimal combination of frequency components for the specific task, thus avoiding local optima. The input to this module is the output from the DFDT module (i.e., the frequency components $Freq$), and the output is the selected frequency components $Freq\_S \in \mathbb{R}^{K\times H'\times W'}$, where K represents the number of selected components.

Through extensive experiments, we discover that the importance of frequency components is quickly determined during the model training process. To exploit this phenomenon and accelerate the frequency component selection process, we design a significance change indicator called Selection Factor Vibration Amplitude (SFVA) to monitor the convergence of the selection factor ranking. Building on this, we propose an early stopping algorithm to accelerate the selection of important frequency components.

In the ACM module, we design multiple 1×1 convolution kernels to convert the obtained $Freq\_S$ to match the input size required by the subsequent backbone model.

Next, we will detail the design of each module.

3.1 DATA FREQUENCY DOMAIN TRANSFORMATION

This section outlines the process of transforming image data to the frequency domain, which is crucial for feature extraction. The DFDT module encompasses the following four steps:

Step 1: RGB to YCrCb. Convert the image from the RGB color space to the YCrCb color space, which separates luminance (Y) from chrominance (Cr and Cb). Post conversion, the image $X_{YCrCb} \in \mathbb{R}^{C\times H\times W}$ is decomposed into three parts: X_Y, X_{Cr}, and X_{Cb}.

Step 2: 8×8 **DCT.** In this step, we transform the image data from the original domain to the frequency domain using DCT. Take X_Y as an example, we apply block-wise DCT to X_Y. The X_Y is divided into multiple non-overlapping image blocks of size 8×8, following the JPEG method (Hudson et al., 2017). Among which, each

blocks of X_Y can be represented by $f_{m,n} \in \mathbb{R}^{8\times8}$, where $m \in \{0, 1, 2, \ldots, H' - 1\}, n \in \{0, 1, 2, \ldots, W' - 1\}$. Then, we perform DCT on $f_{m,n}$.

Step 3: Frequency Combination. After DCT, we obtain $H' \times W'$ DCT coefficient matrices $F_{m,n} \in \mathbb{R}^{8\times8}$, each containing 64 DCT coefficients, where $H' = H/8$, and $W' = W/8$. In this step, we combine coefficients at the same position from different DCT matrices (i.e., $F_{m,n}[x, y]$) into single-frequency component (i.e., $Freq\_Y^t$, where $t \in \{0, 1, 2, \ldots, 63\}$). After combining all frequency component matrices, concatenate them along the component dimension to obtain frequency component tensor $Freq\_Y$.

Step 4: Concatenate. After performing Steps 2 and 3 on channels Y, Cr, and Cb, we concatenate the resulting tensors $Freq\_Y$, $Freq\_Cr$, and $Freq\_Cb$ along the component dimension to form the final frequency component tensor $Freq \in \mathbb{R}^{C'\times H'\times W'}$.

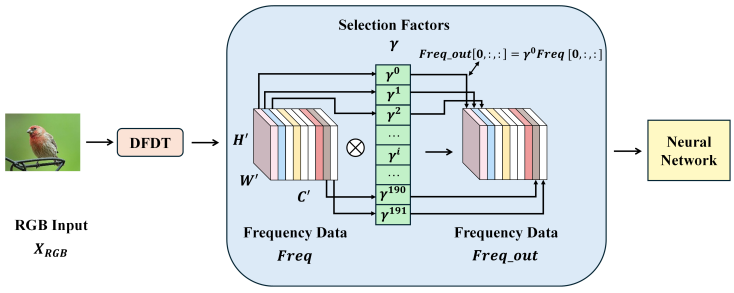
For more details on this section, please refer to Appendix A.1.

3.2 Frequency Component Selection

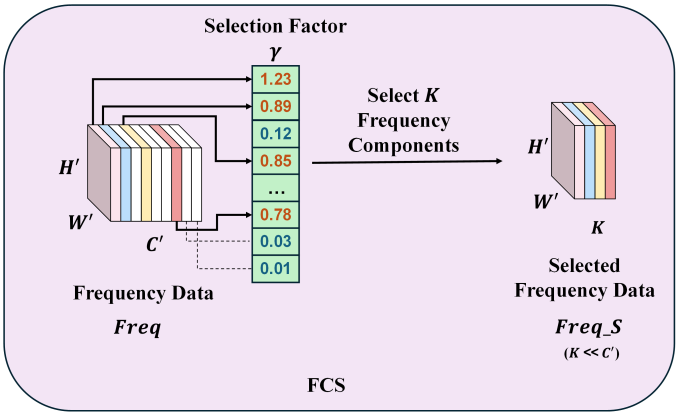
In Section 3.1, we obtain the frequency component tensor $Freq$ to represent the RGB image data X_{RGB} in the frequency domain. Each slice $Freq[i, :, :]$ corresponds to a specific frequency component of X_{RGB}, in this section, we aim to select the optimal subset from $Freq$.

Research has shown (El Qacimy et al., 2014; Fu & Guimaraes, 2016) that different frequency components can have varying impacts on the performance of DL models. Therefore, to effectively extract crucial features and minimize redundancy in frequency-domain data, selecting the most relevant frequency components is essential.

As discussed in Section.2, the existing methods often select the fixed frequency components that are not tailored for specific downstream tasks which may result in suboptimal model performance. Although some other studies propose the learning-based method to select frequency components by greedy strategy, they still suffer from the local optima and high computation cost.



(a)



(b)

Figure 2: Frequency Component Selection

Design of the Selection Factor Layer. To adaptively select the optimal frequency components for various tasks, we add a selection factor layer between the DFDT module and the subsequent DL model. Thus, we can minimize modifications to subsequent DL model as much as possible, achieving plug-and-play functionality. The selection factor layer is illustrated in Fig. 2. This layer includes a learnable parameter for each frequency component, denoted as selection factor γ^i. This results in selection factors $\gamma \in \mathbb{R}^{C'}$, which have the same size (i.e., C') with the number of frequency components in $Freq$.

With the selection factor layer, each component of $Freq$ is scaled by its corresponding selection factor:

$$Freq\_out[i, :, :] = \gamma^i Freq[i, :, :], \quad (1)$$

where $i \in \{0, 1, 2, \ldots, C' - 1\}$. $Freq\_out$ represents the output after applying the selection factors.

The magnitude of γ^i directly affects $Freq\_out[i, :, :]$. A smaller γ^i results in a smaller $Freq\_out[i, :, :]$, indicating that the corresponding frequency component is less significant for the subsequent task. Thus, γ^i reflects the importance of each frequency component.

If γ^i approaches zero, the corresponding frequency component $Freq\_out[i, :, :]$ becomes negligible, allowing us to remove it. The sparsity of γ ensures that only the most impactful components are retained.

Training of γ. To ensure that the selected frequency components adapt to various tasks, we train γ jointly with the subsequent DL model. We apply L_1 regularization to enforce sparsity in γ, ensuring that only the most important components are selected. The loss function is formulated as follows:

$$\arg\min_{\theta,\gamma}(\mathcal{L}_{\theta,\gamma}(X) + \lambda||\gamma||_1), \tag{2}$$

where $\mathcal{L}_{\theta,\gamma}(*)$ represents the loss function for the subsequent task, θ denotes the parameters of the task model, λ is a hyperparameter for regularization, X is the input (e.g., X_{RGB}), and $||\cdot||_1$ denotes L_1 regularization.

During training, Eq. (2) serves as the loss function. The selection factors γ are trained alongside the task model parameters θ, allowing the importance of frequency components to be determined by the task.

Selection of Frequency Components. After training, the selection factors γ reflect the importance of each frequency component. We select the K largest values of γ^i and record their indices in the set Ω. The selected frequency components are then:

$$\begin{aligned} &Freq\_S = Freq[\text{index}, :, :], \\ &\text{s.t. index} \in \Omega. \end{aligned} \tag{3}$$

As illustrated in Fig. 2b, the frequency components corresponding to the largest γ values are selected to form a new tensor $Freq\_S \in \mathbb{R}^{K\times H'\times W'}$.

Our method offers the following advantages:

(1) **Adaptive to subsequent tasks.** Unlike fixed selection methods, our approach is data-driven, allowing the importance of frequency components to be determined by the specific task through the training of γ.

(2) **Global optima.** By selecting the best frequency components simultaneously, our method avoids the local optima problem associated with greedy approaches. The well-designed selection factors ensure an optimal combination of frequency components.

(3) **Plug-and-play functionality.** Due to the well-designed selection factor layer between the DFDT module and subsequent DL model, our FCS module is easy to implement and can be easily integrated into various subsequent tasks.

3.3 Early Stopping: Get γ Quickly

While the method described in Section 3.2 effectively identifies critical frequency components, it requires the model to converge, resulting in significant time overhead. However, our selection process focuses on the relative importance of each selection factor rather than their final values. By sorting the selection factors in descending order and selecting the top K, we can determine the set Ω of indices corresponding to these factors. If we can ascertain Ω before the model fully converges, we can expedite the frequency selection process, avoiding prolonged training.

Rapid Convergence of Selection Factor Ranking. Our method updates γ each training epoch to evaluate the relative importance of frequency components and select the K most significant ones. Fig. 3 illustrates the evolution of all selection factors during training, with epochs on the horizontal axis and selection factor values on the vertical axis.

As shown in Fig. 3, the selection factors are categorized into two groups: unselected (brown in legend) and selected (other colors in legend). From the outset of training, the values of selected factors are markedly higher than those of unselected factors. The unselected factors' values decline rapidly early in training and remain near zero, indicating the minimal contribution of their corresponding frequency components to subsequent tasks. Since Ω is the index set of the top K selection factors, its elements stabilize early in training and remain unchanged. For example, if $K = 6$, we finally select the 0-th, 1-th, 64-th, 65-th, 128-th, and 129-th frequency components, result in $\Omega = \{0, 1, 64, 65, 128, 129\}$. We can find that the Ω can be determined after 6 epochs and no longer changes in the following epochs, due to the selection factors γ^0, γ^1, γ^{64}, γ^{65}, γ^{128}, and γ^{129} are significantly larger than other selection factors.

Therefore, if we can detect this stabilization epoch, we can halt the training of γ at that point, to accelerate the selection of frequency component.

Designing the Early Stopping Mechanism. To leverage this rapid convergence phenomenon and accelerate the training process of γ, we introduce a metric to determine whether Ω has converged and propose an early stopping mechanism based on this metric.

The metric, called Selection Factor Vibration Amplitude (SFVA), is used to monitor the convergence of the selection factor ranking, and it is defined as:

$$SFVA_i^K = \left|\bigcup_{j=0}^{N-1} \Omega_{i-j}^K\right| - K, \tag{4}$$

where N is a hyperparameter representing the number of consecutive rounds to consider (the value of N will be discussed in the experimental section), Ω_{i-j}^K denotes the index set corresponding to the top K selection factors in the $(i-j)$-th iteration, $\bigcup_{j=0}^{N-1} \Omega_{i-j}^K$ represents the union set of top K indexes from iteration $i-(N-1)$ to iteration i, and $|\cdot|$ denotes the cardinality of the union set. The i-th iteration represents the current iteration. When calculating the SFVA for the i-th iteration, we also take into account the Ω^K from the previous $(N-1)$ iterations. This can reduce the impact of noise and outliers that may arise from a single iteration, thereby improving the stability of the results.

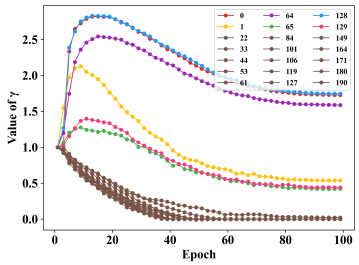


Figure 3: The trend of γ on RESISC45.

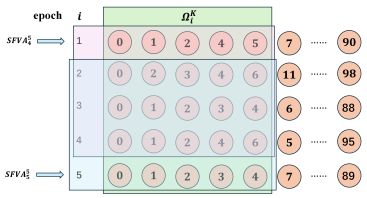


Figure 4: In each epoch, the selection factors...

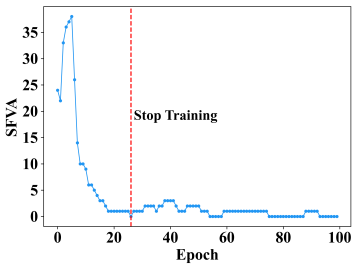


Figure 5: The training period is greatly shortened.

For example, as shown in Fig. 4, with $K = 5$ and $N = 4$, each row represents the index sequence obtained by sorting γ in descending order in the i-th iteration, and each row in the green box is the selected index set Ω_i^K. From these iterations, we obtain five Ω sets: $\Omega_1^5 = [0,1,2,4,5]$, $\Omega_2^5 = [0,2,3,4,6]$, $\Omega_3^5 = [0,1,2,3,4]$, $\Omega_4^5 = [0,1,2,4,6]$, $\Omega_5^5 = [0,1,2,3,4]$. Therefore, we can calculate $SFVA_4^5$ and $SFVA_5^5$ as follows:

$$\begin{aligned} SFVA_4^5 &= \left|\Omega_1^5 \cup \Omega_2^5 \cup \Omega_3^5 \cup \Omega_4^5\right| - K \\ &= |\{0,1,2,3,4,5,6\}| - 5 = 2, \end{aligned} \qquad \begin{aligned} SFVA_5^5 &= \left|\Omega_2^5 \cup \Omega_3^5 \cup \Omega_4^5 \cup \Omega_5^5\right| - K \\ &= |\{0,1,2,3,4,6\}| - 5 = 1. \end{aligned} \tag{5}$$

The SFVA value indicates the degree of change in the elements of Ω. A smaller SFVA value signifies a more stable Ω. Therefore, our early stopping mechanism is: **when SFVA $= 0$, it indicates that Ω is identical across the consecutive N epochs, thereby triggering early stopping of training; otherwise, training continues.**

As shown in Fig. 5, without early stopping, the model is typically trained for a full 100 epochs. By implementing the early stopping mechanism, we utilize the SFVA metric to automatically monitor the training progress of γ, allowing the network to stop training at the appropriate epoch based on SFVA. The red line in the figure indicates the early stopping epoch. With early stopping, we can limit training to approximately 20 epochs, significantly reducing the training time for γ and enabling fast selection of frequency components.

Discussion on N. The choice of N is crucial for the effectiveness of the early stopping mechanism. If N is too small, the SFVA metric may not capture sufficient stability in the selection factors, leading to premature stopping and potentially selecting suboptimal frequency components. Conversely, if N is too large, the early stopping mechanism may not trigger soon enough, thereby reducing the benefit of accelerated training.

In our experiments, we empirically determine the optimal value of N by evaluating the trade-off between training time and the stability of Ω. We find that an N value in the range of 4 to 5 often provides a good balance, ensuring that Ω has stabilized while still achieving significant reductions in training time. This range allows the SFVA metric to effectively monitor the convergence of the selection factors and trigger early stopping at an appropriate point.

3.4 Adaptive Channel Matching

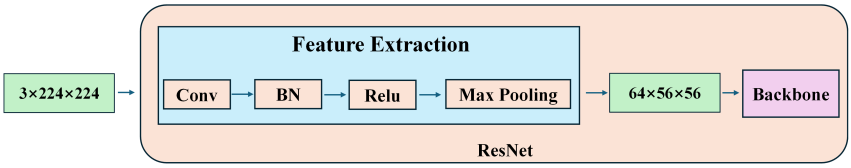
Classic subsequent DL models typically use multi-layer neural networks for feature extraction, then input the feature data X into the backbone model. As shown in Fig.6a, this is a typical example of using ResNet to process data. It takes a $3 \times 224 \times 224$ RGB image as input, and then uses the input layer(Conv + BN + ReLU + MaxPool) to perform simple feature extraction to obtain a tensor of size $64 \times 56 \times 56$, which is the size accepted by the backbone network of ResNet.

In this paper, we want to use the existing backbone model for subsequent tasks, which requires that the size of our frequency features match the input size required by the backbone model. In our AFFS algorithm, we perform feature extraction in the frequency domain using the DFDT and FCS modules to obtain $Freq\_S \in \mathbb{R}^{K \times H' \times W'}$. Then we design an ACM module to transform $Freq\_S$ into input that matches the existing model.

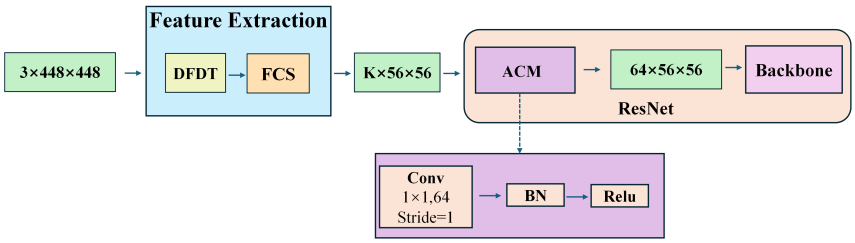
Specifically, in the ACM module, we mainly design a 1×1 convolution layer in which the number of convolution kernels is the same as the number of input channels required by the backbone model, followed by a BN layer and a ReLU layer. This allows us to feed the extracted frequency features into the existing model.

Note that, in our method, larger images can be used as input. We randomly crop and scale the original image to a larger size (e.g., 448×448) instead of 224×224, allowing us to obtain more information from an image.

The pseudocode and execution process of this paper can be found in Appendix A.2.



(a) Traditional ResNet



(b) ResNet with ACM module

Figure 6: (a) Traditional ResNet. (b) ResNet with ACM module.

4 Experiment

We conduct extensive experiments to evaluate our proposed method, addressing the following research questions:

RQ1: How do parameters K and N impact classification accuracy and early stopping?

RQ2: How does our method compare to spatial domain methods and other frequency selection methods?

RQ3: How effective is the early stopping mechanism?

RQ4: How effective are the selection factors?

RQ5: Is our method a general feature extraction technique that has robust effectiveness across various DL models?

For details on the dataset and implementation, please refer to Appendix A.3, A.4.

4.1 Comparison of Classification Accuracy with Other Methods

Table 1: Precision comparison of different methods on three datasets. Best results are highlighted in bold.

Model	Backbone	Domain	Input Size	Acc		
				CIFAR10	ImageNet20	RESISC45
ResNet	ResNet-18	RGB	$3 \times 224 \times 224$	92.10	81.65	95.11
SENet		RGB	$3 \times 224 \times 224$	92.37	81.93	95.53
FCANet		RGB	$3 \times 224 \times 224$	92.40	81.97	95.55
DCTNet-192		Frequency	$192 \times 56 \times 56$	92.45	81.82	95.26
DCTNet-24		Frequency	$24 \times 56 \times 56$	92.66	82.95	95.45
AFFS-24 (ours)		Frequency	$24 \times 56 \times 56$	**93.06**	**83.56**	**95.89**
ResNet	ResNet-50	RGB	$3 \times 224 \times 224$	92.43	81.73	95.35
SENet		RGB	$3 \times 224 \times 224$	92.61	81.95	95.67
FCANet		RGB	$3 \times 224 \times 224$	92.66	82.03	95.71
DCTNet-192		Frequency	$192 \times 56 \times 56$	91.59	81.89	95.43
DCTNet-24		Frequency	$24 \times 56 \times 56$	92.87	82.65	95.78
AFFS-24 (ours)		Frequency	$24 \times 56 \times 56$	**93.11**	**83.67**	**96.04**
DenseNet	DenseNet-121	RGB	$3 \times 224 \times 224$	92.45	84.81	96.01
DCTNet-192		Frequency	$192 \times 56 \times 56$	92.65	84.84	96.12
DCTNet-24		Frequency	$24 \times 56 \times 56$	92.85	85.49	96.44
AFFS-24 (ours)		Frequency	$24 \times 56 \times 56$	**93.21**	**85.87**	**96.72**

In this section, we conduct experiments on three datasets (e.g., CIFAR10, ImageNet20, RESISC45) and three backbone models (e.g., ResNet-18(He et al., 2016), ResNet-50(He et al., 2016), DenseNet-121(Huang et al., 2017)).

To answer **RQ2**, we implement the following classification methods on ResNet18 and ResNet50: SENet(Hu et al., 2018), FCANet(Qin et al., 2021), DCTNet-192(Qin et al., 2021), DCTNet-24(Qin et al., 2021), and our AFFS-24.

In addition, we also implement DCTNet-192(Qin et al., 2021), DCTNet-24(Qin et al., 2021), and our AFFS-24 methods on Densenet121.

Among these, the ResNet is the classic residual network model, and the DenseNet is a variant of ResNet which connects each layer to all preceding layers, resulting in a network with more connections and more comprehensive information flow. The SENet enhances ResNet by adding a weight to each convolutional layer's kernel to implement channel-wise attention learning and the FCANet uses DCT weight coefficients as the weight of each channel. These three models usually input the image data in the original spatial domain rather than transform it to the frequency domain. Differently, the DCTNet-192 and DCTNet-24 take frequency domain data as input and propose a gate module to select 192 and 24 frequency components, respectively.

Comparison with Other Classification Methods. As shown in Table 1, the column Domain indicates whether spatial domain data or frequency domain data is input to the network, RGB indicates spatial domain data, and Frequency indicates frequency domain data. The column Input Size indicates the size of the data input into the model. The experimental results in Table 1 shows:

Accuracy: The size of the feature data extracted by our AFFS reduced by nearly 90% of the original frequency feature, thereby decreasing the data size input to the subsequent backbone models, this means that even if we input much smaller data feature, we can improve the model accuracy by about 1% across all experiments.

These results demonstrate that processing data in the frequency domain, rather than the spatial domain, more effectively extracts key information, reduces data volume, and enables the model to better extract features, thereby improving classification accuracy.

Comparison with Fixed Frequency Selection. To validate the effectiveness of our frequency selection strategy using γ, we compare it with two fixed frequency selection modes: Square and Triangle. In square selection mode, we select the K lowest frequency components for each component of Y, Cr, and Cb, respectively. These components are concentrated in the upper left corner of the DCT coefficients and arranged in a rectangular shape. The triangle selection mode is similar, except that the selected components are arranged in a triangular shape in the DCT coefficient matrix. These specific selection modes are illustrated in Fig. 7.

Table 2: Accuracy comparison between different frequency selection modes. Best results are highlighted in bold.

Selection Mode	Dataset	Number of Components (K)			
		27	30	45	48
Square	CIFAR10	92.19	—	—	92.12
Triangle		—	92.23	92.31	—
Ours		**92.97**	**92.91**	**92.87**	**92.85**
Square	ImageNet20	82.60	—	—	82.27
Triangle		—	81.45	82.05	—
Ours		**83.35**	**83.30**	**82.85**	**82.83**
Square	RESISC45	95.15	—	—	95.11
Triangle		—	95.38	95.29	—
Ours		**95.73**	**95.69**	**95.55**	**95.51**

The experimental results are presented in Table 2. It can be seen that when the number of selected frequency components is the same, our method leads to higher model accuracy in all cases. Although both fixed selection patterns choose low-frequency components that contain the most information, they overlook the varying importance of different frequencies for the subsequent tasks. Our method, by using γ to represent the significance of each frequency component, precisely selects the most relevant frequencies for the task at hand as γ is trained with the task. This adaptive approach allows the model to retain or discard frequency components based on task-specific requirements, thereby optimizing input features and enhancing classification performance.

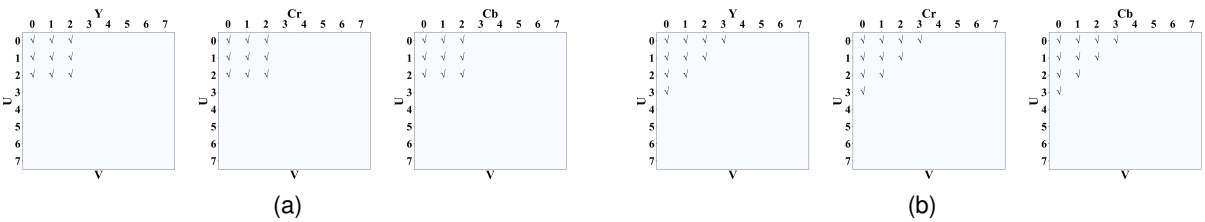


Figure 7: (a) Square selection mode. (b) Triangle selection mode.

Selected Frequency Components by AFFS. To understand the frequency components selected by our method, we visualize the distribution of frequencies in the DCT coefficient matrix for $K = 24$. The results are shown in Fig. 12. Frequencies marked with ✓ indicate the components selected by our method. Observations include:

The selected frequencies show similar distributions across the three datasets, with some variations in individual frequency selections.

The majority of selected frequencies are located in the top-left corner of the DCT coefficient matrix, representing low-frequency components. However, for the Y component, some mid-to-high-frequency components in the bottom-left and top-right corners are also chosen, this means that these mid-to-high-frequency components have better performance in specific tasks, but they are roughly discarded in the fixed selection method.

Across all datasets, the model prefers selecting more Y components, which represent the luminance information of the image. The Cr and Cb components are chosen less frequently, which represent the chrominance information of the image. This mirrors human visual perception, which is more sensitive to luminance information, suggesting that neural networks also extract features more effectively from luminance information.

These findings indicate that in image classification tasks, luminance information contributes more significantly, and low-frequency components, which carry richer information, are predominantly retained by the network. Our approach demonstrates the ability to dynamically select frequencies based on data-driven methods across various datasets. The number and positions of selected Y, Cr, and Cb components are different in datasets, allowing the model to adapt its selection strategy according to dataset characteristics, thereby maximizing classification performance. This dynamic selection enables the model to better capture and utilize key information in images, leading to improved classification accuracy.

4.2 The Other Experimental Results.

To answer **RQ1**, we conduct some experiments to understand the influence of the parameter K on classification accuracy and early stopping. Those results indicate that training neural networks based on features extracted and selected in the frequency domain effectively supports image classification tasks. An optimal number of frequency components yield better performance than baseline models. A consistent pattern across the datasets shows that more frequency components do not necessarily lead to better performance, highlighting the importance of selecting the right number of components to maximize accuracy and achieve significant data compression. Balancing the reduction of training epochs with improving accuracy is crucial to selecting an appropriate N. From the experimental results, setting N to 4 or 5 achieves an optimal balance, reducing training time for γ by about 80% without a significant drop in accuracy. In this paper, we choose $N = 5$. For more details on the experiment, please refer to Appendix A.5.

To answer **RQ3**, we evaluate the impact of indexes selected by γ using the early stopping mechanism compared to full training. Experiments were conducted on the RESISC45 dataset with three different backbone networks. The experimental results confirm that the effectiveness of the early stopping mechanism, indicating that useful γ values can be obtained with fewer iterations. Table 4 also highlights the stability of γ, as the indexes selected for the same dataset remain consistent across different models with minor fluctuations. For more details on the experiment, please refer to Appendix A.6.

To answer **RQ4**, we validate the effectiveness of the γ proposed in this paper by conducting ablation experiments. The results show two major advantages of selection factors. First, each frequency component is weighted by selection factor to enhance useful information and suppress invalid information. Even if we do not make any selections, the model accuracy improves. Secondly, the selection factor reflects the importance of each frequency component, so we can select the most important subset of $Freq$ according to it. This can remove a lot of redundant information and even improve accuracy. For more details on the experiment, please refer to Appendix A.7.

RQ5 is answered in Appendix A.4.

5 Conclusion

To extract more effective data features for different subsequent DL models in the frequency domain, we propose a novel approach for task-specific frequency component selection in the context of deep learning. By designing the selection factor layer, we can dynamically identify optimal frequency component combinations for different subsequent tasks. Moreover, we propose the SFVA metric to effectively monitor the convergence of the selection factor ranking, and further propose an early stopping mechanism to accelerates the selection of frequency components, which significantly reduces training time but maintaining or even improving model performance. Extensive experimental results demonstrate that our method achieves higher subsequent DL model accuracy compared to other frequency component selection methods, while significantly improving selection speed.

References

Nasir Ahmed, T_ Natarajan, and Kamisetty R Rao. Discrete cosine transform. *IEEE transactions on Computers*, 100(1):90–93, 1974.

Mario Barbero, H Hofmann, and ND Wells. Dct source coding and current implementations for hdtv. *EBU Technical Review*, 251:22–33, 1992.

J Bharadiya. Convolutional neural networks for image classification. *International Journal of Innovative Science and Research Technology*, 8(5):673–677, 2023.

Ronald N Bracewell. The fourier transform. *Scientific American*, 260(6):86–95, 1989.

Jireh Yi-Le Chan, Khean Thye Bea, Steven Mun Hong Leow, Seuk Wai Phoong, and Wai Khuen Cheng. State of the art: a review of sentiment analysis based on sequential transfer learning. *Artificial Intelligence Review*, 56 (1):749–780, 2023.

Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 105(10):1865–1883, 2017.

Amandeep Singh Dhanjal and Williamjeet Singh. A comprehensive survey on automatic speech recognition using neural networks. *Multimedia Tools and Applications*, 83(8):23367–23412, 2024.

Samuel Felipe dos Santos, Nicu Sebe, and Jurandy Almeida. The good, the bad, and the ugly: Neural networks straight from jpeg. In *2020 IEEE International Conference on Image Processing (ICIP)*, pp. 1896–1900. IEEE, 2020.

Hatice Vildan Dudukcu, Murat Taskiran, Zehra Gulru Cam Taskiran, and Tulay Yildirim. Temporal convolutional networks with rnn approach for chaotic time series prediction. *Applied soft computing*, 133:109945, 2023.

Bouchra El Qacimy, Mounir Ait Kerroum, and Ahmed Hammouch. Feature extraction based on dct for handwritten digit recognition. *International Journal of Computer Science Issues (IJCSI)*, 11(6):27, 2014.

Dan Fu and Gabriel Guimaraes. Using compression to speed up image classification in artificial neural networks. *Technical report*, 2016.

Lionel Gueguen, Alex Sergeev, Ben Kadlec, Rosanne Liu, and Jason Yosinski. Faster neural networks straight from jpeg. *Advances in Neural Information Processing Systems*, 31, 2018.

Shengxi Gui, Shuang Song, Rongjun Qin, and Yang Tang. Remote sensing object detection in the deep learning era—a review. *Remote Sensing*, 16(2):327, 2024.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Graham Hudson, Alain Léger, Birger Niss, and István Sebestyén. Jpeg at 25: Still going strong. *IEEE MultiMedia*, 24(2):96–103, 2017.

Jaeho Jeon, Seongyong Lee, and Hohsung Choe. Beyond chatgpt: A conceptual framework and systematic review of speech-recognition chatbots for language learning. *Computers & Education*, pp. 104898, 2023a.

Jaeho Jeon, Seongyong Lee, and Seongyune Choi. A systematic review of research on speech-recognition chatbots for language learning: Implications for future directions in the era of large language models. *Interactive Learning Environments*, pp. 1–19, 2023b.

Sudarsana Reddy Kadiri, Paavo Alku, and Bayya Yegnanarayana. Analysis of instantaneous frequency components of speech signals for epoch extraction. *Computer Speech & Language*, 78:101443, 2023.

Masaya Kawamura, Yuma Shirahata, Ryuichi Yamamoto, and Kentaro Tachibana. Lightweight and high-fidelity end-to-end text-to-speech with multi-band generation and inverse short-time fourier transform. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2023.

Lingshun Kong, Jiangxin Dong, Jianjun Ge, Mingqiang Li, and Jinshan Pan. Efficient frequency domain-based transformers for high-quality image deblurring. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5886–5895, 2023.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Jinyu Li et al. Recent advances in end-to-end automatic speech recognition. *APSIPA Transactions on Signal and Information Processing*, 11(1), 2022.

Pingli Ma, Chen Li, Md Mamunur Rahaman, Yudong Yao, Jiawei Zhang, Shuojia Zou, Xin Zhao, and Marcin Grzegorzek. A state-of-the-art survey of object detection techniques in microorganism image analysis: from classical methods to deep learning approaches. *Artificial Intelligence Review*, 56(2):1627–1698, 2023.

José Maurício, Inês Domingues, and Jorge Bernardino. Comparing vision transformers and convolutional neural networks for image classification: A literature review. *Applied Sciences*, 13(9):5521, 2023.

Mohammad Amin Morid, Olivia R Liu Sheng, and Joseph Dunbar. Time series prediction using deep learning methods in healthcare. *ACM Transactions on Management Information Systems*, 14(1):1–29, 2023.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Badri N Patro, Vinay P Namboodiri, and Vijay Srinivas Agneeswaran. Spectformer: Frequency and attention is what you need in a vision transformer. *arXiv preprint arXiv:2304.06446*, 2023.

Zequn Qin, Pengyi Zhang, Fei Wu, and Xi Li. Fcanet: Frequency channel attention networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 783–792, 2021.

Bulla Rajesh, Mohammed Javed, Shubham Srivastava, et al. Dct-compcnn: A novel image classification network using jpeg compressed dct coefficients. In *2019 IEEE Conference on Information and Communication Technology*, pp. 1–6. IEEE, 2019.

Daniele Ravì, Miroslaw Bober, Giovanni Maria Farinella, Mirko Guarnera, and Sebastiano Battiato. Semantic segmentation of images exploiting dct based features and random forest. *Pattern Recognition*, 52:260–273, 2016.

Li Ruan, Yu Bai, Shaoning Li, Shuibing He, and Limin Xiao. Workload time series prediction in storage systems: a deep learning based approach. *Cluster Computing*, pp. 1–11, 2023.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

Kai Xu, Minghai Qin, Fei Sun, Yuhao Wang, Yen-Kuang Chen, and Fengbo Ren. Learning in the frequency domain. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1740–1749, 2020.

Shige Xu, Lei Zhang, Yin Tang, Chaolei Han, Hao Wu, and Aiguo Song. Channel attention for sensor-based activity recognition: embedding features into all frequencies in dct domain. *IEEE Transactions on Knowledge and Data Engineering*, 35(12):12497–12512, 2023.

Shen Yan, Xuehan Xiong, Anurag Arnab, Zhichao Lu, Mi Zhang, Chen Sun, and Cordelia Schmid. Multiview transformers for video recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3333–3343, 2022.

Dengsheng Zhang and Dengsheng Zhang. Wavelet transform. *Fundamentals of image data mining: Analysis, Features, Classification and Retrieval*, pp. 35–44, 2019.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*, pp. 27268–27286. PMLR, 2022.

A Appendix

A.1 DFDT

Step 1: RGB to YCrCb. Convert the image from the RGB color space to the YCrCb color space, which separates luminance (Y) from chrominance (Cr and Cb). Post conversion, the image $X_{YCrCb} \in \mathbb{R}^{C\times H\times W}$ is decomposed into three parts:

$$\begin{aligned} X_Y &= X_{YCrCb}[:,:,0] \in \mathbb{R}^{H\times W}, \\ X_{Cr} &= X_{YCrCb}[:,:,1] \in \mathbb{R}^{H\times W}, \\ X_{Cb} &= X_{YCrCb}[:,:,2] \in \mathbb{R}^{H\times W}. \end{aligned} \tag{6}$$

Since the operations are identical for each channel, we'll use the Y channel as an example for the following steps.

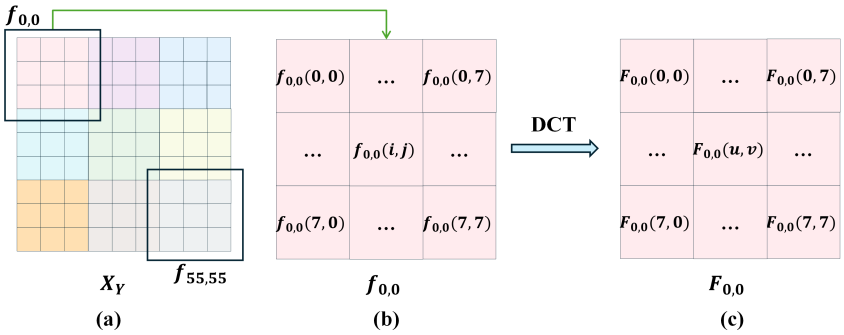


Figure 8: 8×8 DCT. (a) X_Y is divided into multiple non-overlapping 8×8 blocks. (b) $f_{0,0}$ is the first image block. (c) $F_{0,0}$ is the coefficient matrix obtained after performing DCT on $f_{0,0}$.

Step 2: 8×8 **DCT.** In this step, we transform the image data from the original domain to the frequency domain using DCT. Specifically, we apply block-wise DCT to X_Y. The X_Y is divided into multiple non-overlapping image blocks of size 8×8, following the JPEG method (Hudson et al., 2017). Among which, each blocks of X_Y can be represented by $f_{m,n} \in \mathbb{R}^{8\times8}$, where $m \in \{0, 1, 2, \ldots, H' - 1\}, n \in \{0, 1, 2, \ldots, W' - 1\}$. Then, we perform DCT on $f_{m,n}$ as follows:

$$F_{m,n}(u,v) = \frac{c_u c_v}{4} \sum_{i=0}^{7} \sum_{j=0}^{7} f_{m,n}(i,j) \cos\left(\frac{(2i+1)u\pi}{2\times8}\right) \cos\left(\frac{(2j+1)v\pi}{2\times8}\right), \quad \text{where } c_u, c_v = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } u, v = 0, \\ 1 & \text{otherwise,} \end{cases} \quad 0 \le u, v \le 7. \tag{7}$$

here, u and v are the horizontal and vertical frequencies, $f_{m,n}(i,j)$ is the pixel value at (i,j), and $F_{m,n} \in \mathbb{R}^{8\times8}$ is the resulting DCT coefficient matrix. The frequency component mentioned in this paper is converted from this coefficient matrix.

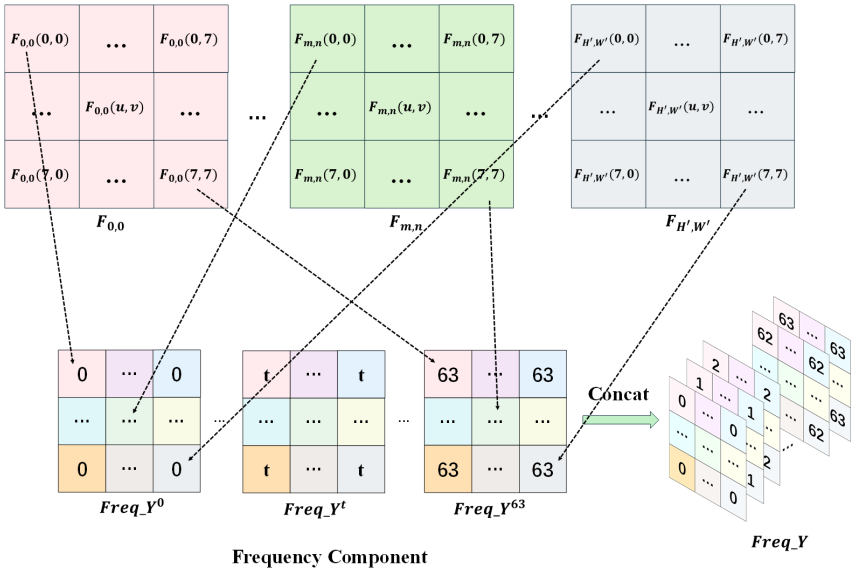


Figure 9: Frequency Combination: Combine frequencies at the same position in each block $F_{m,n}$ to form a frequency component matrix $Freq\_Y^t$. Finally, concatenate all matrices to obtain the tensor $Freq\_Y$.

Step 3: Frequency Combination. After DCT, we obtain $H' \times W'$ DCT coefficient matrices $F_{m,n} \in \mathbb{R}^{8\times8}$, each containing 64 DCT coefficients, where $H' = H/8$, and $W' = W/8$.

In this step, we combine coefficients at the same position from different DCT matrices (i.e., $F_{m,n}[x,y]$) into single-frequency component (i.e., $Freq\_Y^t$, where $t \in \{0, 1, 2, \ldots, 63\}$) as follows:

$$
\begin{aligned}
&Freq\_Y^t[i,j] = F_{m,n}[x,y], \\
&\text{s.t. } x = \lfloor t/8 \rfloor \ , \ y = t\%8, \\
&i \in \{0,1,2,\ldots,H'-1\}, \\
&j \in \{0,1,2,\ldots,W'-1\}.
\end{aligned} \tag{8}
$$

After combining all frequency component matrices by Eq.(8), concatenate them along the component dimension to obtain frequency component tensor $Freq\_Y$:

$$
Freq\_Y = \text{Concat}([Freq\_Y^0, \ldots, Freq\_Y^t, \ldots, Freq\_Y^{63}]). \tag{9}
$$

Similarly, $Freq\_Cr$ and $Freq\_Cb$ are obtained by:

$$
\begin{aligned}
Freq\_Cr &= \text{Concat}([Freq\_Cr^0, \ldots, Freq\_Cr^t, \ldots, Freq\_Cr^{63}]), \\
Freq\_Cb &= \text{Concat}([Freq\_Cb^0, \ldots, Freq\_Cb^t, \ldots, Freq\_Cb^{63}]).
\end{aligned} \tag{10}
$$

Step 4: Concatenate. After performing Steps 2 and 3 on channels Y, Cr, and Cb, we concatenate the resulting tensors $Freq\_Y$, $Freq\_Cr$, and $Freq\_Cb$ along the component dimension to form the final frequency component tensor $Freq \in \mathbb{R}^{C' \times H' \times W'}$, as illustrated in Fig. 10.

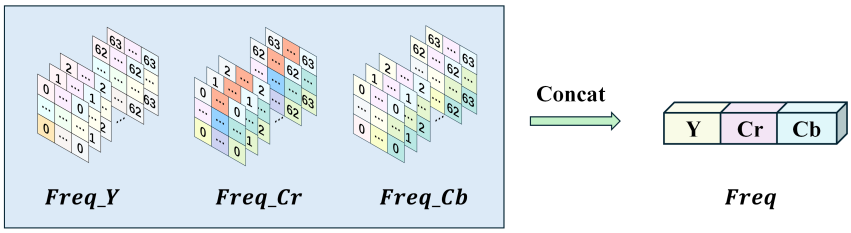


Figure 10: Concatenate $Freq\_Y$, $Freq\_Cr$, and $Freq\_Cb$ along the component dimension to form the tensor $Freq$ with a size of $192 \times H/8 \times W/8$, where $192 = (64 + 64 + 64)$.

A.2 Execution Process

As previously discussed, DL-based approaches primarily involve two stages: feature extraction and inputting the extracted feature data into subsequent DL model. Our AFFS algorithm adheres to this two-step process, as depicted in Algorithm 1.

In lines $4 - 14$, we concentrate on the feature extraction stage. Initially, we transform the input data X into the frequency domain using the DFDT module (line 4). Subsequently, the FCS module is employed to select the optimal frequency components (denoted as $Freq\_S$) for the subsequent DL model (lines $5 - 15$). During this phase, an early stopping mechanism is utilized to expedite the frequency component selection, as outlined in lines $7 - 13$.

Upon selecting the frequency components, we obtain the extracted features $Freq\_S$ in the frequency domain. We then move to the second stage: inputting the extracted features $Freq\_S$ into the subsequent DL model for fine-tuning and final output (lines $15 - 17$). To facilitate this, the ACM module is applied to adjust the dimensions of $Freq\_S$ to meet the dimension requirements of the subsequent DL model.

Although we use image classification as the task to illustrate our algorithm, AFFS is a general-purpose algorithm that can efficiently and effectively select frequency components that are sensitive to various deep learning tasks.

A.3 Datasets

Our method is a general frequency-based feature extraction technique designed to enhance DL model's performance by leveraging a smaller subset of selected frequency features. To evaluate its effectiveness, we implement our method in conjunction with a multi-class image classification task using three real-world datasets, described as follows:

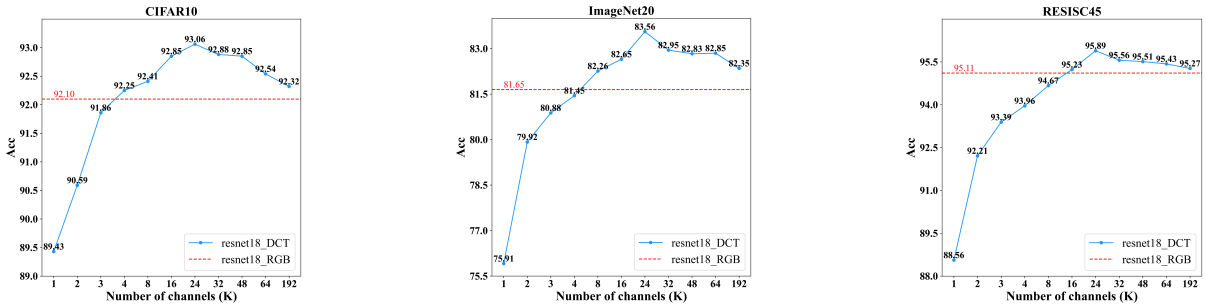
CIFAR10 (Krizhevsky et al., 2009): This dataset comprises 60,000 RGB images distributed across 10 categories, including airplanes, cars, and birds. Each category contains 6,000 images, with 50,000 images allocated for training and 10,000 for testing. The images have a resolution of 32×32.

ImageNet (Russakovsky et al., 2015): The original ImageNet1000 dataset contains 1.2 million training images, 50,000 validation images, and 10,000 test images, spanning 1,000 categories. For our experiments, we randomly selected 20 categories, using 500 images per category for training and 100 images per category for testing. The average image size in this dataset is 482×415.

Algorithm 1 AFFS

Input: X, N, K
Output: Trained Model
Begin
Transform X to $Freq$ using DFDT
for epoch from 1 to 100 **do**
  Train Model $\mathcal{L}_{\theta,\gamma}$ using Eq. (2), obtain θ, γ
  // Start early stopping detection
  Calculate $SFVA$ using Eq. (4)
  if $SFVA == 0$ **then**
    break // Early Stopping
  end if
  // Complete early stopping detection
end for
Select $Freq\_S$ from $Freq$ using Eq. (3)
Match $Freq\_S$ with the model's required input using ACM
Fine-tune the model with the selected frequency components
Return the trained model
End

NWPU-RESISC45 (Cheng et al., 2017): This remote sensing image dataset covers 45 scene categories, each containing 700 images, totaling 31,500 images. We used 27,000 images for training and 4,500 images for testing. The images have a resolution of 256×256. Hereafter, we refer to this dataset as RESISC45.



(a) (b) (c)

Figure 11: Accuracy with different numbers of components.

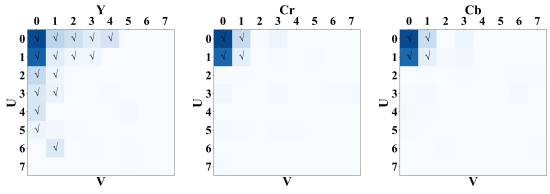
A.4 Implementation Details

Our method is a versatile technique with the potential to positively impact various DL models. To demonstrate this and answer **RQ5**, we implement the image classification task in three different ways by using three widely adopted CNNs as backbone models: ResNet18 (He et al., 2016), ResNet50 (He et al., 2016), and DenseNet121 (Huang et al., 2017). All models are implemented using the PyTorch (Paszke et al., 2019) framework and are trained on a single NVIDIA GeForce RTX 3090 GPU.

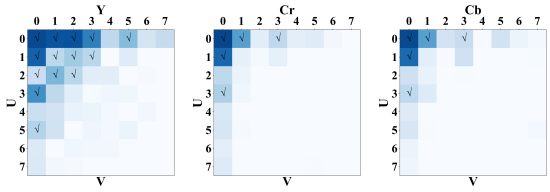
We employ the SGD optimizer with a momentum of 0.9 and a weight decay of 0.0001. During the training of the selection factors γ, the batch size is set to 16, the initial learning rate is 0.01, and the regularization parameter λ is 0.0002. The selection factors γ are initialized as a vector of all ones. For fine-tuning with the selected critical data, the batch size is increased to 64, and the initial learning rate is 0.1. All training employs cosine learning rate decay and label smoothing. We train all models from scratch for 100 epochs.

The loss function $\mathcal{L}_{\text{loss}}$ used in this study is the cross-entropy loss, which is directly related to accuracy. We use accuracy (Acc) as the evaluation metric, which measures the proportion of predictions where the model's predicted class matches the true class. Specifically, it assesses the proportion of instances where the model's highest probability prediction corresponds to the true label, reflecting the model's ability to correctly identify the single most likely class.

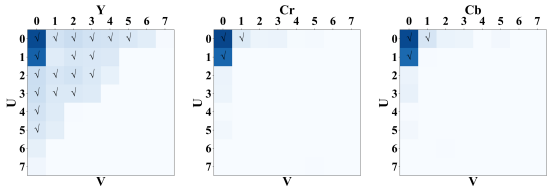
Due to the nature of the DCT, the values between different frequency components can vary significantly. Therefore, we pre-compute the mean and variance of each component across all training data and standardize each component accordingly.



(a) CIFAR10.



(b) ImageNet20.



(c) RESISC45.

Figure 12: Distribution of selected components in Y, Cr, Cb ($K = 24$).

A.5 The Impact of Parameters K and N

To answer **RQ1**, we conduct the following experiments to understand the influence of the parameter K on classification accuracy and early stopping.

Impact of K on Classification Accuracy. The parameter K denotes the number of frequency components selected from $Freq$. Using the selection method described in Section 3.2, we chose the K most relevant components out of 192 available frequency components for training and inference. Fig. 11 displays the results across three datasets, revealing consistent patterns:

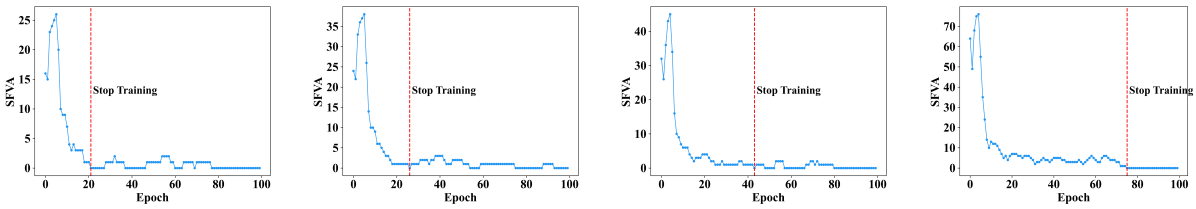
- **Single Component (Freq-0):** When only the first frequency component is used for training, reducing the data dimension from $3 \times 448 \times 448$ to $1 \times 56 \times 56$, the model still performs well. This is due to the DCT's energy concentration property, which compresses most of the signal's energy into the DC component at frequency 0, retaining significant information from the original image.
- **Increasing Components:** As the number of selected components increases, the input information becomes richer, leading to improved accuracy. Peak accuracy is observed at 24 components. Beyond this point, additional components may introduce redundant data or noise, causing a slight decrease in accuracy.
- **Optimal Components:** Using the top 24 components results in the best model accuracy, reducing the size of the data input into the subsequent DL model by nearly 90% compared to using all 192 frequency components. Despite the reduced input size, improvements of 0.96%, 1.91%, and 0.78% in classification accuracy are observed. This demonstrates that appropriate frequency component selection can largely reduce the original frequency feature size to 10% while enhancing model performance.

These results indicate that training neural networks based on features extracted and selected in the frequency domain effectively supports image classification tasks. An optimal number of frequency components yield better performance than baseline models. A consistent pattern across the datasets shows that more frequency components do not necessarily lead to better performance, highlighting the importance of selecting the right number of components to maximize accuracy and achieve significant data compression.

Impact of K on Early Stopping. To understand K's effect on early stopping, we conducted experiments using ResNet18 on ImageNet20. Fig. 13 shows the results:

- **Training Time:** As K increases, the early stopping mechanism's stopping time is delayed, leading to longer training times for γ.

- **Balance:** While a smaller K shortens training time, it should not be too small to avoid reducing accuracy and causing early stopping, which leads to insufficient training of γ. Balancing model accuracy and training time, we set $K = 24$ as the optimal value.



(a) $K = 16$ (b) $K = 24$ (c) $K = 32$ (d) $K = 64$

Figure 13: Effect of parameter K on early stopping.

The Impact of N on Classification Accuracy and Early Stopping. N is a parameter in SFVA, representing the number of consecutive epochs considered in our analysis. We evaluated its impact using ResNet18 on the ImageNet20 and RESISC45 datasets. The experimental results, summarized in Table 3, reveal the following insights:

- **Smaller N:** When N is smaller, training stops earlier, resulting in less effective γ. This leads to lower accuracy after fine-tuning based on γ. An insufficiently trained γ results in suboptimal frequency component selection due to inaccurate rankings.
- **Larger N:** As N increases, more training epochs are considered to evaluate the stable status of the order of frequency factors, allowing γ to undergo more extensive training. This results in more accurate combinations of frequency components. However, excessively large N values increase training epochs significantly, defeating the purpose of early stopping.

Balancing the reduction of training epochs with improving accuracy is crucial to selecting an appropriate N. From the experimental results, setting N to 4 or 5 achieves an optimal balance, reducing training time for γ by about 80% without a significant drop in accuracy. In this paper, we choose $N = 5$.

Table 3: The impact of parameter N on training epochs and accuracy ($K = 24$).

N	2	3	4	5	6	7
ImageNet20 (Epoch/Acc)	14/82.35	15/82.35	20/83.51	28/83.49	29/83.49	56/83.52
RESISC45 (Epoch/Acc)	6/94.76	13/95.42	15/95.81	19/95.85	27/95.84	36/95.86

A.6 Effect of Early Stopping Mechanism

To answer **RQ3**, we evaluate the impact of indexes selected by γ using the early stopping mechanism compared to full training. Experiments were conducted on the RESISC45 dataset with three different backbone networks. Table 4 summarizes the results, revealing:

• In all models, after adding the early stopping mechanism, the models stop training early after 20 epochs of training. Compared with the case without early stopping, the training process of selection factors is shortened by about 80%, which demonstrates the mechanism's effectiveness across different architectures.

• There is minimal difference in selected indexes between full training and early stopping. Only a few mid-range frequencies differ, which has a negligible impact on accuracy.

These findings confirm the effectiveness of the early stopping mechanism, indicating that useful γ values can be obtained with fewer iterations. Table 4 also highlights the stability of γ, as the indexes selected for the same dataset remain consistent across different models with minor fluctuations.

A.7 Ablation Study

To answer **RQ4**, we validate the effectiveness of the γ proposed in this paper by conducting ablation experiments on the following three aspects:

1. "192 components, without γ": The obtained $Freq$ by DFDT is directly input into the subsequent DL model.

Table 4: Impact of Early Stopping ($N = 5, K = 24$).

Backbone	Early Stopping	Training Epoch	Training Time	Top24-indexes	Acc
ResNet18	NO	100	33457s	0, 1, 2, 3, 4, 5, 8, 10, 11, **16**, 17, 18, 19, 24, 25, **26**, 32, 40, 64, 65, 72, 128, 129, 136	95.89
	YES	19	6230s	0, 1, 2, 3, 4, 5, 8, 10, 11, **15**, 17, 18, 19, 24, 25, **27**, 32, 40, 64, 65, 72, 128, 129, 136	95.85
ResNet50	NO	100	35192s	0, 1, 2, 3, 4, 5, 8, 9, **11**, 12, 16, 17, 18, **19**, 24, 25, 32, 40, 64, 65, 72, 128, 129, 136	96.04
	YES	21	6659s	0, 1, 2, 3, 4, 5, 8, 9, **10**, 12, 16, 17, 18, **20**, 24, 25, 32, 40, 64, 65, 72, 128, 129, 136	95.97
DenseNet121	NO	100	39684s	0, 1, 2, 3, 4, 5, 8, **10**, **12**, 16, 17, 18, 19, 24, 25, 26, 32, 40, 64, 65, 72, 128, 129, 136	96.72
	YES	17	6605s	0, 1, 2, 3, 4, 5, 8, **9**, 16, 17, 18, 19, 24, 25, 26, **27**, 32, 40, 64, 65, 72, 128, 129, 136	96.64

2. "192 components, with γ": The obtained $Freq$ is first processed through the selection factor layer to obtain $Freq\_out$ (as shown in Eq. 1), and then $Freq\_out$ is input into the DL model.

3. "24 components, selected by γ": The obtained $Freq$ is precisely selected into 24 components according to γ ($Freq\_S(K = 24)$) and then input into the DL model.

Table 5 presents the detailed results:

• "192 components, without γ": The model achieves the lowest classification accuracy among all settings.

• "192 components, with γ": Introducing γ to scale the frequency components enhances effective information and suppresses noise, resulting in improved accuracy compared with that "192 components, without γ".

• "24 components, selected by γ": Selecting key frequency components via γ optimizes model performance by retaining critical information and discarding noise, achieving the best results.

The results in Table 5 show two major advantages of selection factors. First, each frequency component is weighted by selection factor to enhance useful information and suppress invalid information. Even if we do not make any selections, the model accuracy improves. Secondly, the selection factor reflects the importance of each frequency component, so we can select the most important subset of $Freq$ according to it. This can remove a lot of redundant information and even improve accuracy.

Table 5: Ablation Experiment.

Ablation	Backbone	Acc		
		CIFAR10	ImageNet20	RESISC45
192 components, without γ	ResNet-18	92.32	82.35	95.27
192 components, with γ		92.69	82.93	95.58
24 components, selected by γ		**93.06**	**83.56**	**95.89**
192 components, without γ	ResNet-50	92.48	82.44	95.54
192 components, with γ		92.79	82.96	95.79
24 components, selected by γ		**93.11**	**83.67**	**96.04**
192 components, without γ	DenseNet-121	92.71	84.12	95.75
192 components, with γ		92.98	85.05	96.32
24 components, selected by γ		**93.21**	**85.87**	**96.72**