Cross-Architecture Feature Alignment with a ResNet-to-ViT Adapter

Cirullo Fabio - Stefan Andrei Alexandru
Università degli Studi di Bari "Aldo Moro"

Abstract

Large Vision-Language Models (VLMs) have demonstrated significant capabilities in understanding and generating content involving both visual and textual modalities. However, their high computational cost, often driven by large Vision Transformer (ViT), poses a significant barrier to deployment on resource-constrained devices. This paper addresses this challenge by proposing a hybrid architecture that replaces the original vision encoder of the Qwen2.5-VL-3B-Instruct model with a more efficient Convolutional Neural Network (CNN), specifically ResNet-50. To bridge the gap between the feature representations of these two distinct architectures, we design and train a lightweight adapter module. This adapter learns to transform the intermediate feature maps from multiple stages of ResNet-50 into a sequence of embeddings that mimics the output of Qwen-VL's original vision encoder. Using features from the ResNet-50, our experiments on the miniImageNet dataset show that this method effectively allows the language model to produce contextually relevant image descriptions. Through adapter training, our work demonstrates that it is possible to combine various vision architectures to create more efficient VLMs.

1 Introduction

In recent years, artificial intelligence has experienced rapid progress in Vision–Language Models (VLMs), which integrate visual and textual information to solve complex tasks. Models like Qwen-VL [1] have set new benchmarks by leveraging large-scale pre-training architectures, often relying on a Vision Transformer (ViT) [2] to encode visual information.

Despite the strong performance of Vision Transformers (ViTs), their computational demands during both training and inference are significant. The self-attention mechanism has a quadratic complexity with respect to the number of input patches, making it a obstacle for applications requiring real-time processing or deployment on edge devices.

In contrast, Convolutional Neural Networks (CNNs) such as ResNet [3] have long served as the foundation of computer vision, valued for their ability to extract hierarchical features efficiently. This work investigates a central question: is it possible to merge the efficiency of a CNN-based vision encoder with the advanced reasoning abilities of a large language model originally trained with a ViT encoder?

Our approach involves substituting the original ViT of the Qwen-VL model with a pre-trained ResNet-50 (as shown in Figure 1). The challenge lies in aligning the feature space of ResNet-50 with the one expected by the Qwen language model. To solve this, we first developed a method for extracting and preparing paired feature sets: intermediate embeddings from ResNet-50 and target embeddings from the original Qwen-VL vision encoder. Using this data, we designed and trained an adapter that learns the mapping between the two visual feature spaces by fusing feature maps from all four stages of the ResNet. The resulting hybrid system allows the ResNet-50 and adapter to provide visual information to the Qwen language model for text generation.

2 Related Work

Our work is situated at the intersection of efficient Vision-Language Model (VLM) design, model adaptation, and cross-modal feature alignment.

Vision-Language Models Modern VLMs, such as BLIP [4] and Qwen-VL [1], typically consist of a vision encoder, a language model, and a mechanism to connect them. A prominent example is LLaVA [5], which connects a pre-trained ViT vision encoder (CLIP) to a large language model (Vicuna). The connection between its vision and language components is a simple linear projection layer. Although effective, this approach does not reduce the high computational demands of the ViT

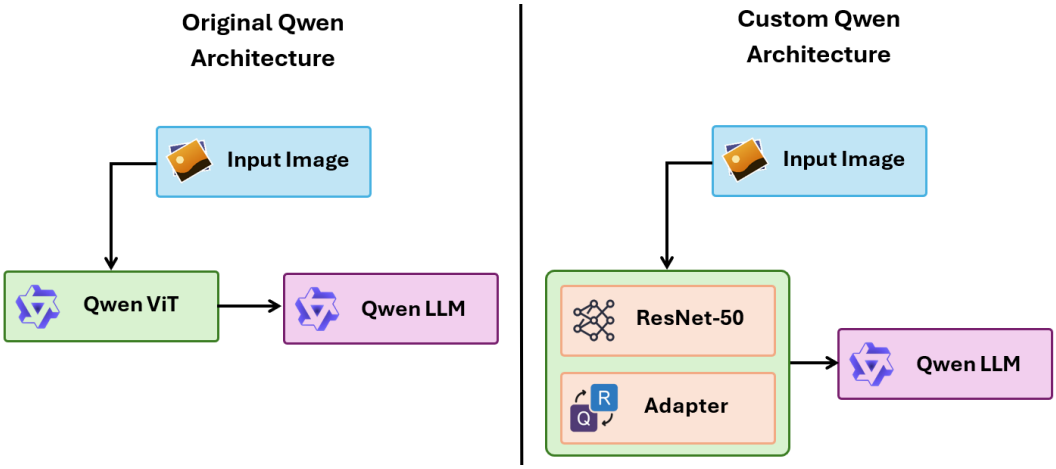

Figure 1: Custom architecture for Qwen model.

backbone, which continue to limit deployment on resource-constrained devices.

Cross-Architecture Adapters Adapters were first introduced as a parameter-efficient fine-tuning (PEFT) technique for large language models [6]. They are small neural network modules that are fine-tuned while the main model's weights remain frozen. This concept has been successfully extended to computer vision. For instance, ViT-Adapter [7] introduces a dedicated, learnable adapter to a plain Vision Transformer to inject CNN-like inductive biases, improving its performance on dense prediction tasks like object detection and segmentation. The idea of using lightweight, CNN-based adapters to augment large Transformer models has proven effective in other domains as well, such as in speech processing [8].

In this project, we extend the adapter concept to a more advanced goal: **inter-architecture feature translation**. Instead of augmenting a ViT, our adapter's goal is to enable the complete replacement of the original ViT with an efficient ResNet-50 encoder. It learns to bridge the feature space gap between these two different architectures, making them compatible.

Multi-Stage Feature Fusion Using features across different layers of a network is a common approach in computer vision. Architectures like Feature Pyramid Networks (FPN) [9] have shown that combining low-level features (rich in spatial detail) with high-level features (rich in semantic content) improves performance. Our adapter draws on all four stages of ResNet, building a rich visual hierarchy that provides the language model with a complete representation. This multi-stage design differs from simpler VLM connections, like the single linear layer in LLaVA [5], which uses only the final output of the vision encoder.

3 Data

The dataset serves as the basis for generating training data for our adapter as well as for the final evaluation. We utilized the **miniImageNet** dataset, a subset of the larger ImageNet dataset [10].

3.1 Dataset Description

The miniImageNet dataset consists of 60,000 color images, each belonging to one of 100 distinct classes, with 600 images per class. The images cover a wide variety of real-world objects and scenes, making it a robust choice for general computer vision tasks. For our project, we used a standard training and test split, resulting in a total of 51,000 images being used.

3.2 Data Preprocessing and Feature Extraction

To train an adapter that maps ResNet features to Qwen-VL features, the first and most important step was generating a parallel dataset of these feature embeddings. This process was split into two main stages.

Image Preprocessing All images in the dataset were processed using a standardized pipeline before being fed into the models, ensuring uniform spatial dimensions of the features. The steps were as follows:

1. **Resizing:** Each image was resized so that its shorter side was 384 pixels. The aspect ratio was maintained.

2. **Center Cropping:** A central crop of 384×384 pixels was taken from each resized image, ensuring that all models receive inputs of the same size.

Generating Target Embeddings (Qwen-VL) Using the original, unmodified Qwen2.5-VL-3B-Instruct model, we processed each of the 51,000 pre-processed images. We isolated the vision encoder part of the model and extracted its output. This output is a tensor of shape (1, 196, 2048), representing the image as a sequence of 196 tokens, each with a dimension of 2048. Each tensor was saved individually under a unique file name, serving as the 'ground truth' for training our adapter.

Generating Source Embeddings (ResNet-50) Next, we used a pre-trained 'microsoft/resnet-50' model to process the same 384x384 images. Instead of just taking the final output, we used hooks to capture the feature maps from the output of each of the four main stages of the ResNet's encoder. This gave us four distinct tensors for each image, with shapes corresponding to the different levels of the feature hierarchy:

- **Stage 0**: '(1, 256, 96, 96)'
- **Stage 1**: '(1, 512, 48, 48)'
- **Stage 2**: '(1, 1024, 24, 24)'
- **Stage 3**: '(1, 2048, 12, 12)'

To avoid the high storage cost and I/O bottlenecks of saving multi-stage ResNet features, we produced them dynamically during training.

4 Methods

The core of our project is to replace the computationally expensive Vision Transformer (ViT) of the Qwen-VL model with a more efficient ResNet-50 encoder. This requires an adapter to translate the feature space of the CNN into a format compatible with the Qwen language model. To find the optimal configuration, we explored four distinct methodological approaches, progressively increasing in complexity. All approaches share a common objective: minimizing the Mean Squared Error (MSE) loss between the adapter's output and the pre-computed ground truth embeddings from the original Qwen-VL vision encoder.

4.1 Frozen ResNet with Single-Embedding Adapters

Our initial experiments focused on establishing a baseline for the feature translation task. In this phase, we kept the pre-trained ResNet-50 weights completely frozen and trained only a lightweight adapter module. A feature of this approach was its simplicity: the adapter received as input only the final embedding from the ResNet-50 encoder, discarding the rich information present in the intermediate layers.

We experimented with several adapter architectures, primarily composed of a sequence of fully connected layers (`nn.Linear`) and non-linear activation functions (e.g., GELU). The goal was to find a minimal yet effective mapping. Variations included the inclusion of convolutional layers (`1x1` and `3x3`), to process the spatial information before flattening, and different numbers of layers. This approach proved to be limited, as the single feature vector from the final ResNet layer lacked the detailed information necessary for a high-fidelity translation.

4.2 Frozen ResNet with Multi-Stage Adapters

After recognizing the limitations of the first approach, we developed a more advanced multi-stage adapter. The idea was to leverage the entire feature hierarchy of the ResNet encoder. In this approach, the ResNet-50 weights remained frozen, but the adapter was designed to accept and process the feature maps from all four of its main stages simultaneously.

The architecture of our multi-stage adapter, and its subsequent versions, follows a specific pipeline:

1. **Per-Stage Projection**: Each of the four input feature maps from ResNet, having different channel dimensions (256, 512, 1024, 2048), is first projected to a unified, high-dimensional space (e.g., 2048 channels) using a separate `1x1` convolutional layer.

2. **Spatial Upsampling**: To enable fusion, the projected feature maps, which have varying spatial resolutions, are all upsampled via bilinear interpolation to match the spatial dimensions of the deepest feature map (e.g., 12x12).

3. **Feature Fusion**: The tensors are concatenated. This combined tensor is then passed through a "fusion block". In early versions, this block consisted of a simple structure with `1x1` convolutions and a GELU activation function. In later versions, this was enhanced with `3x3` convolutions to better capture local spatial context within the fused feature map.

Layer (type:depth-idx)	Input Shape	Output Shape
MultiStageAdapter	[1, 256, 96, 96]	[1, 196, 2048]
├─ModuleList: 1-1	--	--
│ └─Conv2d: 2-1	[1, 256, 96, 96]	[1, 2048, 96, 96]
│ └─Conv2d: 2-2	[1, 512, 48, 48]	[1, 2048, 48, 48]
│ └─Conv2d: 2-3	[1, 1024, 24, 24]	[1, 2048, 24, 24]
│ └─Conv2d: 2-4	[1, 2048, 12, 12]	[1, 2048, 12, 12]
├─Sequential: 1-2	[1, 8192, 12, 12]	[1, 2048, 12, 12]
│ └─Conv2d: 2-5	[1, 8192, 12, 12]	[1, 512, 12, 12]
│ └─GELU: 2-6	[1, 512, 12, 12]	[1, 512, 12, 12]
│ └─Conv2d: 2-7	[1, 512, 12, 12]	[1, 512, 12, 12]
│ └─GELU: 2-8	[1, 512, 12, 12]	[1, 512, 12, 12]
│ └─Conv2d: 2-9	[1, 512, 12, 12]	[1, 2048, 12, 12]
├─TransformerEncoder: 1-3	[1, 144, 2048]	[1, 144, 2048]
│ └─ModuleList: 2-10	--	--
│ │ └─TransformerEncoderLayer: 3-1	[1, 144, 2048]	[1, 144, 2048]
│ │ └─TransformerEncoderLayer: 3-2	[1, 144, 2048]	[1, 144, 2048]
├─AdaptiveAvgPool2d: 1-4	[1, 2048, 12, 12]	[1, 2048, 14, 14]
├─LayerNorm: 1-5	[1, 196, 2048]	[1, 196, 2048]

Figure 2: Architecture of Adapter

4. **Sequence Generation**: The final step converts the fused 2D feature map into a 1D sequence of tokens suitable for the language model. This process was significantly refined across versions:
   - In our baseline model, the pipeline is: **Adaptive Pooling** to a fixed grid size (e.g., 14x14), **Flattening** the grid into a sequence (e.g., 196 tokens), **Adding** learnable positional embeddings.
   - In our most advanced versions, we introduced a **Transformer Encoder** layer. This self-attention mechanism is applied directly to the fused feature map before the adaptive pooling step, allowing the model to learn global relationships between different spatial regions of the image, resulting in representations that capture long-range relationships within the data.
5. **Final Normalization**: A final `LayerNorm` is applied to stabilize the output before it is passed to the language model. This step is consistent across all versions of the adapter.

To identify the most effective architecture of the multi-stage adapter, we incrementally added complexity to it. The tested versions can be summarized as follows:

Version 1 (Baseline) This model implements our multi-stage approach using a lightweight fusion block based on `1x1` convolutions, followed by adaptive pooling and learnable positional embeddings to produce the output sequence.

Version 2 (Self-Attention) This version introduced a **Transformer Encoder** after the fusion block. The goal was to capture global dependencies between image regions through self-attention before the pooling step.

Version 3 (Enhanced Local Context) This version enhanced the fusion block of the baseline model by incorporating a `3x3` **convolutional layer**. This change was designed to improve the adapter's ability to learn local spatial patterns.

Version 4 (Hybrid Approach) Our most advanced model (shown in Figure 2) combines the features of the previous iterations. It uses both the enhanced fusion block with `3x3` convolutions from version 3 and the Transformer Encoder for global self-attention from version 2, creating a comprehensive architecture that processes both local and global context.

4.3 Joint Fine-tuning of ResNet and Adapter

We hypothesized that fine-tuning the ResNet encoder for our task would help better align its feature representations with the target feature space. Therefore, in our third approach, we unfroze the weights of the ResNet-50 encoder and trained them jointly with the Multi-Stage Adapter.

This fine-tuning was configured with a different learning rate. We typically used a smaller learning rate for the ResNet backbone (e.g., `LEARNING_RATE_RESNET = 1e-4`) and a larger one for the adapter (`LEARNING_RATE_ADAPTER = 1e-3`), since the adapter was being trained from scratch while the ResNet only required minor adjustments. This strategy allows the pre-trained CNN to adapt to the task avoiding significant

forgetting of previously learned representations, while the adapter learns the complex mapping more rapidly.

4.4 Sequential Fine-tuning (ResNet then Adapter)

Our final approach was a two-step process, designed to achieve maximum specialization. This method extends the model obtained from the previous approach.

1. **Load Best Jointly-Tuned Model**: We started by loading the weights of the best performing model from the joint fine-tuning phase, where both the ResNet and the adapter had been updated.
2. **Freeze ResNet and Fine-tune Adapter**: We then froze the now-specialized weights of the ResNet-50 encoder. With a fixed, high-quality feature source, we proceeded to fine-tune only the adapter for a few additional epochs with a smaller learning rate.

The idea is that once the ResNet has adapted its feature hierarchy, the adapter can benefit from a final training phase to perfect its translation mapping based on this new, more stable input distribution.

5 Experiments

The final phase of our project was to evaluate the performance of our best-trained components. The evaluation consisted of two parts: we conducted quantitative tests to measure the precision of our feature space translation and performed qualitative inference to assess the real-world performance of the final hybrid model. This section describes the inference setup and presents the results from both our quantitative analysis, based on the Mean Squared Error (MSE) on the test set, and our qualitative evaluations.

5.1 Quantitative Results

To evaluate the effectiveness of our proposed methods, we first trained different adapter architectures using the training set and monitored the validation loss to select the best-performing models. Subsequently, we performed a final evaluation of these selected models on a test set to measure their generalization capability. All training, validation, and testing were performed on splits of the miniImageNet dataset.

For the final evaluation, we used Mean Squared Error (MSE) on the test set to measure how closely the embeddings produced by our adapter match the ground truth embeddings from the original Qwen-VL vision encoder. Table 5 reports the test loss and standard deviation for the best model of each of the four approaches described above.

The results show a clear improvement across the different strategies. The single-embedding adapters (Approach 1) struggled to capture the required complexity, resulting in the highest MSE loss. Introducing the multi-stage architecture (Approach 2) significantly reduced the loss, highlighting the benefit of leveraging ResNet's full feature hierarchy. Jointly fine-tuning the ResNet backbone (Approach 3) further lowered the loss and its variance. Finally, the sequential fine-tuning strategy (Approach 4) achieved the best results with a test loss of **2.3569** and the lowest standard deviation, demonstrating the most accurate and stable feature space translation.

5.2 Inference Process

The inference process involves replacing the vision encoder of the Qwen-VL model with our custom-built composite model (ResNet-50 and adapter).

1. **Model Assembly:**
   - We instantiate the full Qwen2.5-VL-3B-Instruct model.
   - We create our composite model, which internally contains the pre-trained ResNet-50 encoder and our trained adapter.
   - We overwrite the original vision encoder of the Qwen model with a simple identity layer. This is because the visual feature extraction is now handled entirely outside the Qwen model by our composite model.
2. **Data Flow for a New Image:**
   - An input image is provided, along with a text prompt (e.g., "Describe this image.").
   - The image is preprocessed to 384×384 using resize and center crop.
   - The preprocessed image is passed through our composite model. This model internally passes it through ResNet-50, extracts the four-stage feature maps, and then feeds these into the trained adapter.
   - The composite model outputs a single tensor of shape (196, 2048).
   - This tensor is manually placed into the input payload that is fed to the Qwen model's 'generate' function, effectively replacing the `pixel_values` that would

Methodological Approach	Adapter Version	Test Loss	Std
1. Frozen ResNet (Single-Embedding)	Various (Linear/CNN)	3.20	-
2. Frozen ResNet (Multi-Stage)	Version 1	2.7226	0.6750
3. Joint Fine-tuning (ResNet + Adapter)	Version 4	2.3967	0.5899
4. Sequential Fine-tuning (Adapter-only)	**Version 4**	**2.3569**	**0.5676**

Table 1: Best validation loss (MSE) achieved for each methodological approach. Lower values indicate a better match with the original Qwen-VL embeddings.

have been processed by the original vision encoder.

- The Qwen language model takes this externally generated visual embedding together with the tokenized text prompt to produce a textual description.

5.3 Qualitative Results

While quantitative metrics confirm the accuracy of our feature space translation, qualitative analysis is needed to evaluate the practical performance and generalization of our hybrid VLM. We tested our best-performing model not only on miniImageNet test images but also on a diverse set of real-world images outside the training distribution. This allowed us to see how well the model describes novel objects and scenes. All tests used the prompt: 'Describe this image.'

5.3.1 Generalization on External Images

The most important test of our approach was its performance on out-of-distribution data. To probe the model's real-world understanding and generalization capabilities, we evaluated it on various images completely unrelated to the miniImageNet dataset. The following examples are representative of its performance.

Figure 3: Test image 1, external to the training dataset.

Generated output: "The image depicts a large, cylindrical structure situated in a natural setting. The structure appears to be made of stone or concrete and has multiple levels with flat roofs. It is surrounded by a well-maintained lawn with pathways leading up to it. In the background, there are rolling hills covered with greenery, suggesting that the location might be in a rural or semi-rural area. The overall scene gives an impression of a historical or architectural site, possibly a monument or a significant building from a past era."

Figure 4: Test image 2, external to the training dataset.

Generated output: "The image shows two white dogs standing on a grassy field with a blurred green background, likely indicating a forest or wooded area. The dogs appear to be of the same breed, possibly a breed like a Poodle or a similar hypoallergenic breed, given their fluffy and dense coat. They have upright ears and a slightly alert expression. The lighting suggests it might be daytime, and the overall scene is serene and natural."

Results analysis: The generated descriptions for these real-world images confirm the model's strong ability to generalize. In both cases, the model correctly comprehends the overall context, describing the environment ("rolling hills", "grassy field"), lighting, and general composition of the scene with high accuracy.

However, analyzing the subject identification

reveals an important insight into the model's behavior. For the image in Figure 3, the model provides a detailed architectural and contextual description of the castle. Rather than identifying the specific landmark, it classifies it using the closest general category available from its training knowledge, such as "a historical or architectural site" or "a monument". This shows that the model understands the object class at a high level, even without recognizing the specific real-world entity.

Conversely, the second example (Figure 4) highlights a limitation. The model misidentifies the sheep as "white dogs". This suggests that when faced with an object that shares visual features with a more common category in its training data (e.g., four-legged, white, fluffy animals), the model defaults to the most probable class it knows.

Our ResNet-adapter model effectively produces semantically rich embeddings for unseen images. Although fine-grained object recognition can be affected by the training data distribution, the model consistently captures and describes the scene accurately.

5.3.2 Performance on In-Distribution Images

While our model demonstrates strong generalization to external images, its performance becomes notably more precise and detailed when evaluated on images from the miniImageNet test set, which share the same distribution as the training data. This indicates that the adapter has successfully learned the specific nuances of the target feature space for familiar object categories. A representative example is shown in Figure 5 using an image of a tank.

Figure 5: Test image from the miniImageNet test set.

> ***Generated output:*** "The image depicts a military tank, specifically an armored vehicle with a turret and a large main gun. The tank is painted in a camouflage pattern, which helps to blend it into the surrounding environment. The vehicle has multiple wheels for mobility, and the structure appears robust and well-armored, designed for combat operations. The background shows a cloudy sky and some greenery, suggesting that the tank might be stationed in a field or a similar outdoor area."

Results analysis: The generated description of the tank is highly accurate. The model does not just provide a general overview but identifies fine-grained, specific details such as the "turret," "large main gun," "camouflage pattern," and "multiple wheels." This level of precision is significantly higher than that observed for the out-of-distribution images, where the descriptions were contextually correct but more generic.

This shows that our ResNet-adapter model not only captures general scene information but also successfully translates the fine-grained features required by the Qwen LLM for precise object recognition. The adapter successfully bridges the architectural gap, allowing the LLM to access the nuanced information extracted by the ResNet encoder, resulting in descriptions that are both contextually aware and highly specific.

6 Conclusion

In this project, we successfully designed an approach to create a more computationally efficient Vision-Language Model. By replacing the native Vision Transformer of Qwen-VL with a ResNet-50 encoder and bridging the architectural gap with a custom-trained multi-stage adapter, we demonstrated the feasibility of a hybrid CNN-Transformer VLM.

Achievements Our primary achievement is the functional pipeline that generates coherent text descriptions from visual inputs processed by our ResNet-adapter module. This confirms that a lightweight adapter can learn a complex mapping between the feature spaces of two architecturally distinct vision encoders.

Limitations The main limitation observed is the accuracy of the generated text. The descriptions obtained for our test cases were not entirely correct, indicating that our adapter has not yet achieved a perfect alignment in translating the embeddings between the two distinct architectures. This suggests

that minimizing MSE loss alone might not be sufficient to capture all the relevant image details that the language model relies on.

Future Work This work establishes a solid foundation and motivates future research aimed at improving alignment and enhancing the fidelity of generated descriptions.

1. **Different Loss Functions:** To improve feature alignment, future work could explore more sophisticated loss functions beyond MSE. A contrastive loss, for instance, could be added to better structure the learned feature space.

2. **Scaling the Training Dataset:** A crucial next step is to train the adapter on a larger dataset, such as the full ImageNet (1000 categories), to enhance the generalization capabilities of the model.

3. **Textual Loss for Training:** A more advanced training strategy would be to incorporate a loss directly from the language model's output. Instead of relying solely on the MSE loss between embeddings, the training objective could include a textual loss. This would involve comparing the text generated using our adapter's output with the ground truth text generated using the original Qwen-VL embedding. This feedback loop would optimize the adapter not just for feature space alignment, but directly for the final task of generating accurate and coherent text.

In conclusion, this project proves that the powerful reasoning capabilities of large language models can be successfully paired with the computational efficiency of classic CNNs, achieving an effective trade-off between performance and model complexity.

References

[1] Jinze Bai et al. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and more. *arXiv preprint arXiv:2308.12966*, 2023.

[2] Alexey Dosovitskiy et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[3] Kaiming He et al. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[4] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pages 12763–12779. PMLR, 2022.

[5] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023.

[6] Neil Houlsby et al. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*. PMLR, 2019.

[7] Zhe Chen, Yuchen Duan, Wenhai Wang, Junjun He, Tong Lu, Jifeng Dai, and Yu Qiao. ViT-Adapter: Vision transformer adapter for dense predictions, 2022.

[8] Wei-Ning Hsu, Anuroop Sriram, Awni Hannun, Vitaliy Liptchinsky, Jiachen Lian, Shuohui Chen, Kushal Lakhotia, T-T. Hsieh, and Abdel rahman Mohamed. CHAPTER: Exploiting convolutional neural network adapters for self-supervised speech models, 2022.

[9] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.