

SUPPLEMENTARY MATERIAL: DESIGNING DEEP LEARNING PROGRAMS WITH LARGE LANGUAGE MODELS

Anonymous authors

Paper under double-blind review

In our supplementary material, we mainly provide exact prompts used in our dataset creation pipeline and benchmark.

1 FULL PROMPTS USED FOR THE DATASET CREATION PIPELINE

1.1 FILTERING NON-AI-RELATED PAPERS

You will be given an abstract of a research paper. Your job is to determine if the paper is related to any research field on artificial intelligence or machine learning. If it is related, only output Y. If it is not related, only output N. Remember to generate only either of Y or N.

1.2 INSTRUCTION DATA AND MULTIPLE-CHOICE QUESTIONS EXTRACTION

You will be given a full text of a research paper. Your job is to extract important information from the paper and create instruction data and multiple choice questions based on the contents. First, for a paper input, extract (1) keywords that express the paper, (2) main contributions of the paper, (3) if a model architecture is suggested as a contribution, diagram-based model architecture, which is expressed by data (put inside ||), model (put inside []), and techniques (put inside <>) (4) benchmark scores for each experiment, and (5) requirements like GPU memory consumption or time. If some information is not available, just skip it, excluding from the output. Here is an example of an extraction:

```
```json
{
 "keywords": ["Computer Vision", "Deep Learning", "ResNet", "Image Classification", "Feature Extraction"],
 "contributions": [
 "C-ResNet, an enhanced ResNet architecture that surpasses existing ResNet variants in image classification tasks",
 "FeatureLoss, a novel loss function that optimizes feature extraction efficiency in deep networks",
 "ResBlock++, a modified residual block in C-ResNet designed to refine feature propagation and gradient flow",
 "Multi-Scale KD, a knowledge distillation method using multiple scales of image features from teacher models to improve student model performance"
],
}
```

```

054 "diagram" (only if available. try to spread all the nodes to
055 the basic units, so that it is much more detailed than the
056 example below. Specify each node's property.): [
057 "<MODEL>X-resnet": {
058 "diagram": "|input_image (224x224x3)|->[conv_block
059 (112x112x64)]->[residual_block_1 (112x112x64)]->[
060 residual_block_2 (56x56x128)]->[
061 global_average_pooling (1x1x128)]->[
062 fully_connected_layer (128)]->[softmax_output (
063 num_classes)]",
064 "property": [
065 "Optimized for image classification tasks.",
066 "Uses skip connections to prevent gradient
067 vanishing and improve training stability.",
068 "Slightly slower inference compared to simple CNNs
069 due to residual connections.",
070 "Increases model depth without significantly
071 increasing the complexity."
072],
073 "<BLOCK>conv_block": {
074 "diagram": "|input_image (224x224x3)|->[conv_layer
075 (112x112x64)]->[batch_norm (112x112x64)]->[
076 relu_activation (112x112x64)]->[max_pooling (56
077 x56x64)]",
078 "property": [
079 "Performs feature extraction by capturing local
080 patterns in the input image.",
081 "Reduces spatial dimensions quickly while
082 retaining essential features.",
083 "Improves feature interpretability by enforcing
084 sparsity through max pooling."
085],
086 "<BLOCK>residual_block_1": {
087 "diagram": "[conv_layer (56x56x64)]->[batch_norm (56
088 x56x64)]->[relu_activation (56x56x64)]->[
089 conv_layer (56x56x64)]->[skip_connection (identity
090 , 56x56x64)]",
091 "property": [
092 "Increases network depth without overfitting,
093 stabilizing gradients during backpropagation
094 .",
095 "Allows information to flow directly through skip
096 connections, improving gradient flow and
097 training speed.",
098 "Decreases interpretability slightly as deeper
099 layers might learn more abstract
100 representations."
101],
102 "<BLOCK>residual_block_2": {
103 "diagram": "[conv_layer (56x56x128)]->[batch_norm (56
104 x56x128)]->[relu_activation (56x56x128)]->[
105 conv_layer (56x56x128)]->[skip_connection (
106 projection, 56x56x128)]",
107 "property": [

```

```

 "Uses projection in the skip connection to match
 dimensions when needed, enabling more
 flexibility in architecture design.",
 "Increases computational cost compared to identity
 skip connections but improves feature
 extraction capability."
]
},
"<LAYER>global_average_pooling": {
 "diagram": "|feature_map (56x56x128)|->|
 spatial_average (1x1x128)|->|global_feature_vector
 (128)|",
 "property": [
 "Reduces the risk of overfitting by summarizing
 features into a single vector.",
 "Improves model generalization for small datasets
 .",
 "Increases inference speed compared to fully
 connected layers by reducing parameters."
]
},
"<LAYER>fully_connected_layer": {
 "diagram": "|global_feature_vector (128)|->|
 fully_connected_neurons (num_classes)|->|
 output_logits (num_classes)|",
 "property": [
 "Transforms the global feature vector into class-
 specific scores.",
 "High flexibility for learning complex decision
 boundaries but increases the number of
 parameters, leading to slower inference in
 large models."
]
},
"<TECHNIQUE>resnet_kd": {
 "diagram": "[resnet_teacher (large)|, resnet_student (
 small)]-><distill_teacher_knowledge_to_student>->[
 student_with_distilled_knowledge (small)]-><fine-
 tune_on_task>",
 "property": [
 "Knowledge distillation helps transfer information
 from a large teacher model to a smaller
 student model.",
 "Increases inference speed on edge devices by
 reducing model size without losing much
 performance.",
 "Can decrease the interpretability of the student
 model as the distilled knowledge is abstract
 and less tied to input-output relationships."
]
},
"<LOSS>kd_loss": {
 "diagram": "$L_{\text{KD}} = \alpha \cdot L_{\text{CE}}(y, p_s) + (1 - \alpha) \cdot T^2 \cdot L_{\text{KL}}(p_t, p_s)$, where L_{CE} is the
 cross-entropy loss, L_{KL} is the
 Kullback-Leibler divergence, p_t and p_s are
 teacher and student outputs, T is the

```

```

162 temperature, and α balances the loss
163 components.",
164 "property": [
165 "Improves the student model's performance by
166 aligning its output distribution with the
167 teacher model.",
168 "Slows down training due to the additional KL
169 divergence loss, but provides better
170 generalization on unseen data."
171]
172 },
173 "scores": [
174 {
175 "model": "X-ResNet",
176 "base_model": "ResNet",
177 "task": "image classification",
178 "pretrain_dataset": "ImageNet",
179 "finetune_dataset": "X-Image",
180 "test_dataset": "CIFAR-10",
181 "metric": "top-1 accuracy (higher is better)",
182 "score_change": "2.5% higher than ResNet-50",
183 "hyperparameters": {
184 "pretrain_epoch": 100,
185 "finetune_epoch": 10,
186 "batch_size": 64,
187 "learning_rate": 1e-4
188 },
189 ...
190],
191 "requirements" (only if available. try to write exact values
192 instead of comparison): {
193 "gpu_memory": "consumes 70GB of VRAM in a single A100",
194 "train_time": "30 hours on a single A100",
195 "inference_time": "10 seconds/token"
196 }
197 ... (below continues with instruction data)
198 ...
199 Remember, the diagram must include specific name of the model,
200 like "ResNet-50 image encoder" instead of "Image encoder". If
201 such specific information is not available, assume from the
202 text.
203 Then, create 5 instruction data (which has a very detailed output)
204 and 5 multiple-choice questions for each contribution,
205 technique, dataset, model, etc. suggested in the paper.
206 Possible topics and corresponding examples (need to be
207 paraphrased in your output) of each instruction data or
208 question are below but not limited to:
209 - description generation: Could you give me an explanation of
210 $model_name$ => $model_name$ is... This model consists of ...
211 This model is known to be perform well on ... because ...
212 Mathematically, it can be expressed as ... (detailed
213 descriptions so on)
214 - combination prediction: How can we combine $model1$ with
215 $model2$ => To combine $model1$ with $model2$, you can apply
the method called ... This method is beneficial in ... because
... Mathematically, it can be expressed as ... (a detailed
description how to do it and reasoning for such choice)

```

```

- property prediction: What are some important properties of
 $model_name$ => Some known properties of $model_name$ are as
 below: ... (detailed known properties of the $model_name$)
- reasoning: What is the reason $technique_name$ is effective for
 $task$? => (a very detailed theoretical explanation if
 available. If not available, saying unknown is fine.)
- mathematical expression: Describe $technique_name$
 mathematically. => $latex expression$, where P is the
 probability distribution... (a very detailed explanation of
 the expressions and each variable if available. Note the
 meaning of each variable must be very accurately stated.)
- name guess: Is there any technique that consists of ... and has
 property ...? => One technique that may satisfy your
 requirements is $technique_name$, which is ... This technique
 consists of ... This technique is useful when ... because ...
 Mathematically, it can be expressed as ... (a very detailed
 description of $technique_name$)

Multiple choice questions include similar contents. Remember that
instruction data and multiple choice questions will be given
to the student without the paper. Thus, you must not include
paper-referencing data or question, like "what is this paper
about?" or "what is the property of the technique mentioned in
the paper?".

For every instruction data or multiple choice question, mention
the source of the sentence(s) in the paper. When using
abbreviations, state full name in the parentheses to prevent
confusion. Your output must be in the JSON format, which looks
like:
json
... (following from above)
ResNet: {
 "instruction_data": [
 {
 "instruction": "Explain how ResNet addresses the
 vanishing gradient problem in deep networks.",
 "output": "ResNet introduces 'skip connections' or
 'residual connections,' which allow the
 network to bypass certain layers by creating
 shortcuts. This helps the gradient flow more
 easily through deeper layers, reducing the
 vanishing gradient problem that occurs during
 backpropagation. By ensuring that the gradient
 does not diminish as it propagates through
 many layers, ResNet enables the training of
 very deep networks with hundreds of layers.",
 "source": "Skip connections are essential to
 ensuring that deeper networks can be trained
 effectively without suffering from vanishing
 gradients.",
 "category": "optimization"
 },
 {
 "instruction": "What role does batch normalization
 play in ResNet's training stability?",
 "output": "Batch normalization is used in ResNet
 to normalize the activations of each layer,
 which helps stabilize the training process. By
 reducing the internal covariate shift, batch

```

```

normalization ensures that the distribution of
layer inputs remains consistent throughout
training. This not only speeds up the training
process but also improves model
generalization by reducing overfitting.
Combined with residual connections, batch
normalization makes training deep networks
like ResNet more reliable and efficient.",
"source": "The importance of batch normalization
in stabilizing and accelerating the training
of deep networks is emphasized.",
"category": "description generation"
},
...
],
"multiple_choice": [
{
"question": "What advantage does ResNet have over
traditional CNNs in terms of depth?",
"choices": [
"A. ResNet can have more layers without
encountering performance degradation.",
"B. ResNet uses fewer parameters than
traditional CNNs.",
"C. ResNet performs better on smaller datasets
due to its architecture.",
"D. ResNet avoids overfitting entirely due to
batch normalization.",
"E. ResNet requires less computational power
to train."
],
"answer": "A. ResNet can have more layers without
encountering performance degradation.",
"source": "The key advantage of ResNet is its
ability to scale to deeper architectures
without suffering from the performance
degradation that traditional CNNs experience
as layers are added.",
"category": "property prediction"
},
{
"question": "What is the technique used to help
ResNet avoid the vanishing gradient problem?",
"choices": [
"A. batch normalization",
"B. data augmentation",
"C. skip connections",
"D. weight decay",
"E. gradient clipping"
],
"answer": "C. skip connections",
"source": "Skip connections in ResNet help address
the vanishing gradient problem by allowing
gradients to flow directly through the network
, bypassing multiple layers when necessary.",
"category": "name guess"
},
...
]

```

```

 },
 ... the same goes for any other contributions proposed in the
 paper.

```

### 1.3 CLASSES/FUNCTION NAME EXTRACTION

You will be given an abstract of a research paper and the names of Python classes and functions utilized to implement the contributions of the paper. Your job is to extract the names of classes and functions that are potentially related to the contributions of the paper. If there seems no potentially relevant classes/functions, you can output an empty list. Your output format must be a JSON format, which looks like:

```

```json
{
  "class": [$class1$, $class2$, ...],
  "function": [$func1$, $func2$, ...]
}
```
if no class/function is relevant:
```json
{
  "class": [],
  "function": []
}
```

```

### 1.4 CONTRIBUTION-CLASS/FUNCTION MAPPING

You are a professional AI programmer. You will be given a model, block, technique, etc. of a deep learning model and potentially relevant code snippets. Your job is to map each model/block/technique/etc. to a corresponding code snippet/class/function name. For example, for an input

<MODEL>XCNN: {'diagram': '|input\_image|->[xcnn\_block]->[softmax\_layers]->|image\_class|', 'property': ['Optimized on image classification', 'Is much more efficient than naive CNN architecture.']}

<BLOCK>xcnn\_block: {'diagram': '[image\_feature\_extractor]->[convolution\_layers]->[gelu\_activation\_function]', 'property': ['Optimizes training and inference speed']}

and code

```

```python
import torch
import torch.nn as nn
import torch.nn.functional as F

# Define the XCNN block
class XCNNBlock(nn.Module):
    def __init__(self, in_channels, out_channels):
        super(XCNNBlock, self).__init__()
        # Image feature extractor (for example, a convolutional layer)
        self.image_feature_extractor = nn.Conv2d(in_channels,
            out_channels, kernel_size=3, stride=1, padding=1)
        # Convolutional layers

```

```

378         self.convolution_layers = nn.Conv2d(out_channels,
379             out_channels, kernel_size=3, stride=1, padding=1)
380         # GELU activation function
381         self.gelu_activation = nn.GELU()
382
383     def forward(self, x):
384         # Forward pass through image feature extractor
385         x = self.image_feature_extractor(x)
386         # Forward pass through convolution layers
387         x = self.convolution_layers(x)
388         # Apply GELU activation function
389         x = self.gelu_activation(x)
390         return x
391
392 # Define the full XCNN model
393 class XCNN(nn.Module):
394     def __init__(self, num_classes, in_channels=3,
395         feature_channels=64):
396         super(XCNN, self).__init__()
397         # XCNN block (e.g., with 3 input channels for RGB images)
398         self.xcnn_block = XCNNBlock(in_channels, feature_channels)
399         # Final softmax layers (fully connected layers for
400             classification)
401         self.fc = nn.Linear(feature_channels * 32 * 32,
402             num_classes) # Adjust size based on input image
403             dimensions
404
405     def forward(self, x):
406         # Pass through XCNN block
407         x = self.xcnn_block(x)
408         # Flatten the output
409         x = x.view(x.size(0), -1) # Flatten to pass into fully
410             connected layer
411         # Final fully connected layer followed by softmax
412         x = F.softmax(self.fc(x), dim=1)
413         return x
414
415 # Example usage
416 if __name__ == "__main__":
417     # Create an instance of the model (assuming 10 image classes)
418     model = XCNN(num_classes=10)
419     # Example input (batch size: 1, 3 channels, 32x32 image)
420     input_image = torch.randn(1, 3, 32, 32)
421     # Forward pass through the model
422     output = model(input_image)
423     print(output)
424
425 """
426 your output must be:
427 ```json
428 {
429     "<MODEL>XCNN": "class XCNN",
430     "<BLOCK>xcnn_block": "class XCNNBlock",
431 }
432 """
433
434 Additionally, consider:
435 If there is no corresponding code snippet, simply output an empty
436 dictionary {}
437
438 Remember that your output must have an accurate JSON format.

```


2 BENCHMARK DETAILS

2.1 FULL PROMPTS USED FOR THE EVALUATION PIPELINE

2.1.1 REQUIREMENT-BASED ARCHITECTURE PROPOSAL

What is the most advanced model for {task_name}, which can be sufficiently trained in {time_constraints} on {gpu_constraints}?

2.1.2 ARCHITECTURE IMPROVEMENT

You will be given the description of a model. Propose two architectural improvements: one for performance boosting and another for efficiency boosting. In addition, write python code or pseudocode with the improvements integrated with the model.

2.1.3 ARCHITECTURE IMPLEMENTATION

Your task is {task_name} on {dataset_name}. You have {gpu_constraints}. Execution time must be less than {time_constraints}. Fortunately, you have an advisor who is an expert in deep learning model architectures. Here is an advise from the advisor: {idea_by_architect}. The performance must be evaluated using {metric_name}, and the resulting {metric_name} must be logged at './metric.log', with only a float value. To first check if your code works, (1) use a very small sliced dataset (2) set epoch to only 1 (3) include early stopping. Cifar-10 dataset is placed at '/workspace/DeepOnes/DeepBench/cifar10/'. The directory looks like:

```
'''
/workspace/DeepOnes/DeepBench/cifar10
|_ cifar-10-batches-py
|_ batches.meta
|_ data_batch_1
|_ data_batch_2
|_ data_batch_3
|_ data_batch_4
|_ data_batch_5
|_ readme.html
|_ test_batch
'''
```

Based on the advise, implement a fully-executable Python code.

Make sure to use GPU. Place your code in

```
```python
'''
.
```

#### 2.1.4 DEBUGGING

```

486 You are a professional AI programmer. You will be given (1) The
487 task that the code is trying to solve (2) The error-causing
488 code (3) The standard output of the execution (4) The standard
489 error of the execution (5) Summarized previous debugging log.
490 Your job is to fix the error-causing code so that the code
491 works correctly. You can run 'pip install' within shell
492 scripts, rewrite python code, or do both to fix the error.
493 Make sure to use GPU. Remember, you must not change the high-
494 level functionality the neural architecture in the code; only
495 implementation fixes for the debugging purpose is allowed.
496 Other codes, such as imports, training, testing, or
497 dataloading, your modification is free. Remember, your output
498 must be the whole debugged code, not a part of it. Never skip
499 the implementation with 'pass' or comments. Shell scripts
500 should be placed within ```sh\n```n, while python codes
501 should be placed within ```python\n```n. Then, summarize the
502 problem and your solution in one line natural language
503 sentence within ```trial\n```n, like ```trial\nSyntax Error
504 -> changed the line 'print("hello world")' to 'print("hello
505 world")', which closes the unclosed parantheses and thereby
506 resolving the syntax error.\n```nIf the same error is being
507 repeated multiple times, you must look for other solutions.
508 (1) task: {task_name} on {dataset_name}. You have {gpu_constraints}
509 }. Execution time must be less than {time_constraints}. The
510 performance must be evaluated using {metric_name}, and the
511 resulting {metric_name} must be logged at './metric.log', with
512 only a float value. To first check if your code works, (1)
513 use a very small sliced dataset (2) set epoch to only 1 (3)
514 include early stopping. Cifar-10 dataset is placed at './
515 workspace/DeepOnes/DeepBench/cifar10/'. The directory looks
516 like:
517 ```
518 /workspace/DeepOnes/DeepBench/cifar10
519 |_ cifar-10-batches-py
520 |_ batches.meta
521 |_ data_batch_1
522 |_ data_batch_2
523 |_ data_batch_3
524 |_ data_batch_4
525 |_ data_batch_5
526 |_ readme.html
527 |_ test_batch
528 ```
529
530 (2) Error-causing code:
531 ```python
532 {error_causing_code}
533 ```
534
535 (3) Standard output:
536 ```
537 {standard_output}
538 ```
539
540 (4) Standard error:
541 ```
542 {standard_error}
543 ```

```

```
(5) Debugging log
{debugging_log}
```

### 2.1.5 GENERATING FULL CODE WITH OPTIMAL HYPERPARAMETERS

You are a professional AI researcher. You will be given a code snippet, which is using a sliced dataset and 1 epoch for testing. Your job is to convert the code to (1) use full datasets and (2) set appropriate hyperparameters for the code, including the number of epochs, based on an educated guess. Except for these two changes, keep everything the same. Remember, your output must be the whole converted code, not a part of it. Never skip the implementation with 'pass' or comments. Place your converted code within ````python\n```\n`. Make sure 2 things below are included in the code with optimized hyperparameters:

1. If the training time is longer than 3600 seconds, stop training and just evaluate based on the trained model.
2. The resulting score must be logged on `./metric.log`.