

ДОДАТОК А

Код реалізації варіаційної квантової голови моделі QuantumTinyViT

```

import torch
import torch.nn as torch_nn
import pennylane as qml

def create_quantum_layer(config: ExperimentConfig):
    quantum_device = qml.device(
        "default.qubit",
        wires = config.number_of_qubits,
    )

    @qml.qnode(quantum_device, interface = "torch")
    def quantum_circuit(inputs, weights):
        qml.AngleEmbedding(
            inputs,
            wires = range(config.number_of_qubits),
            rotation = "Y",
        )

        qml.StronglyEntanglingLayers(
            weights,
            wires = range(config.number_of_qubits),
        )

        return [
            qml.expval(qml.PauliZ(wire))
            for wire in range(config.number_of_qubits)
        ]

weight_shapes = {
    "weights": (
        config.quantum_depth,

```

```

        config.number_of_qubits,
        3,
    )
}

```

```

quantum_layer = qml.qnn.TorchLayer(
    quantum_circuit,
    weight_shapes,
)

```

```

return quantum_layer

```

```

class QuantumTinyViT(torch.nn.Module):
    def __init__(self, config: ExperimentConfig):
        super().__init__()

        self.backbone = TinyVisionTransformerBackbone(config)

        self.feature_projection = torch.nn.Linear(
            config.embedding_size,
            config.number_of_qubits,
        )

        self.quantum_layer = create_quantum_layer(config)

        self.classifier = torch.nn.Linear(
            config.number_of_qubits,
            config.number_of_classes,
        )

    def forward(self, images: torch.Tensor) -> torch.Tensor:
        class_features = self.backbone(images)

        quantum_inputs = self.feature_projection(class_features)

```

```
quantum_outputs = self.quantum_layer(quantum_inputs)

logits = self.classifier(quantum_outputs)

return logits
```

ДОДАТОК Б

Відомість кваліфікаційної роботи

Позначення					Найменування		Дод. відомості	
					Текстові документи			
1.					Пояснювальна записка		73 с.	