```python
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

# Configuration
image_size = 64
patch_size = 8
num_patches = (image_size // patch_size) ** 2  # 64 patches
embedding_dim = 64
num_heads = 4
num_layers = 2

# Build Vision Transformer
def create_vit_model():
    inputs = layers.Input(shape=(image_size, image_size, 3))

    # 1. Create patches
    patches = layers.Conv2D(
        embedding_dim,
        kernel_size=patch_size,
        strides=patch_size,
        padding='valid',
        name='patch_embedding'
    )(inputs)

    # 2. Reshape patches
    patches = layers.Reshape((num_patches, embedding_dim), name='reshape_patches')(patches)

    # 3. Add positional embeddings
    positions = tf.range(start=0, limit=num_patches, delta=1)
    pos_embedding = layers.Embedding(
        input_dim=num_patches,
        output_dim=embedding_dim,
        name='positional_embedding'
    )(positions)
    patches = patches + pos_embedding

    # 4. Transformer blocks
    for i in range(num_layers):
        # Multi-head self-attention
        x1 = layers.LayerNormalization(name=f'norm1_layer{i}')(patches)
        attention = layers.MultiHeadAttention(
            num_heads=num_heads,
            key_dim=embedding_dim,
            name=f'attention_layer{i}'
```

```
        )(x1, x1)
        x2 = layers.Add(name=f'add1_layer{i}')([patches, attention])

        # Feed-forward network
        x3 = layers.LayerNormalization(name=f'norm2_layer{i}')(x2)
        x3 = layers.Dense(embedding_dim * 2, activation='relu', name=f'ffn_dense1_layer{i}')
        x3 = layers.Dense(embedding_dim, name=f'ffn_dense2_layer{i}')(x3)
        patches = layers.Add(name=f'add2_layer{i}')([x2, x3])

    # 5. Global average pooling
    representation = layers.GlobalAveragePooling1D(name='global_pooling')(patches)

    # 6. Classification head
    features = layers.Dense(64, activation='relu', name='pre_classifier')(representation)
    features = layers.Dropout(0.5, name='dropout')(features)
    outputs = layers.Dense(1, activation='sigmoid', name='gender_output')(features)

    model = keras.Model(inputs=inputs, outputs=outputs, name='vision_transformer')
    return model

# Create and save the model
vit_model = create_vit_model()
vit_model.save('gender_classifier_vit.h5')

# Also save as TensorFlow SavedModel format (often shows more detail in Netron)
vit_model.save('gender_classifier_vit_savedmodel')

print("Model saved! You can now upload either file to netron.app")
vit_model.summary()
```