Model Comparison on Single Image

This notebook compares the predictions of three trained models—Base MobileNetV2, Fine-Tuned MobileNetV2, and a custom CNN—on the same test image.

In [28]:
```
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.models import load_model
import tensorflow as tf
from tensorflow.keras.models import Model
import cv2
```

In [3]:
```
# Load the three trained models
base_model = load_model("mobilenetv2_base_model.h5")
fine_tuned_model = load_model("mobilenetv2_finetuned_model.h5")
cnn_model = load_model("cnn_stroke_model.keras")
```

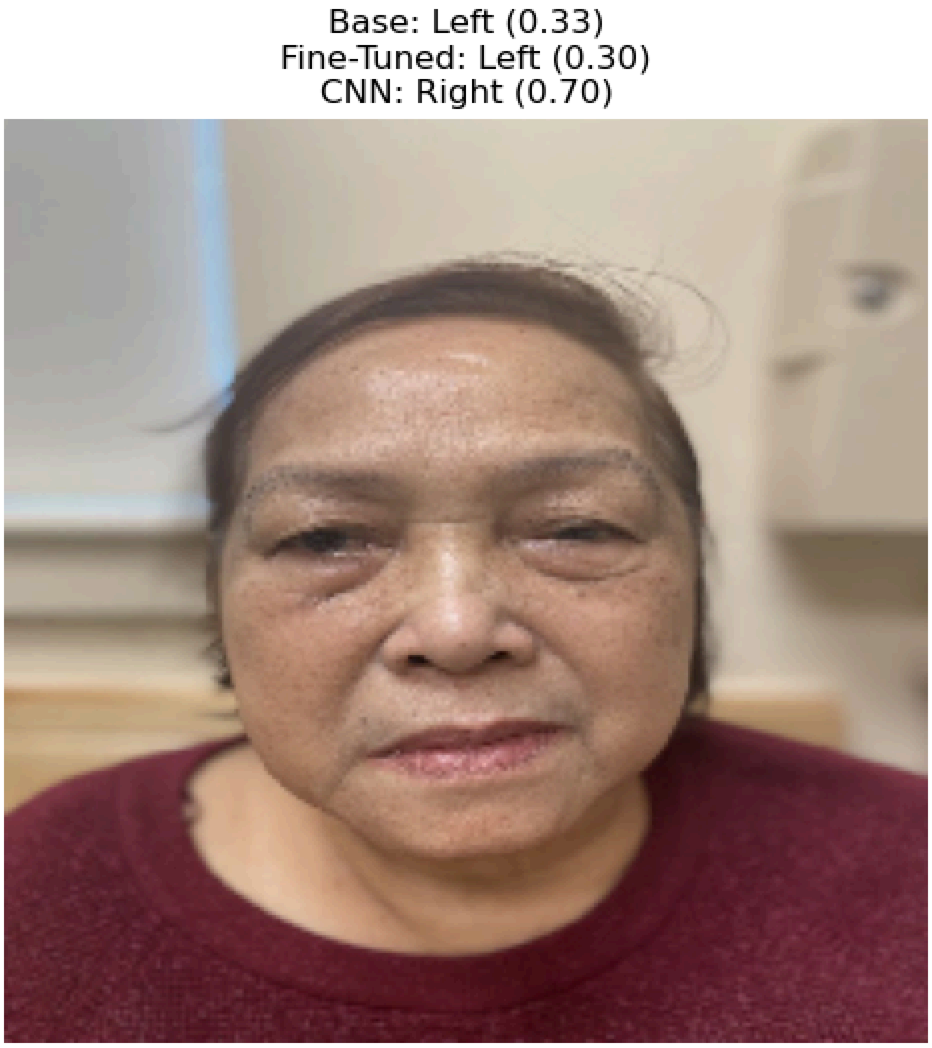
```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to
be built. `model.compile_metrics` will be empty until you train or evaluate
the model.
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to
be built. `model.compile_metrics` will be empty until you train or evaluate
the model.
```

In [22]:
```
# Load the image (PIL format)
img_path = "test_image.jpg"  # <-- replace with your actual file
img = load_img(img_path, target_size=(224, 224))

# Convert to array and normalize
img_array = img_to_array(img) / 255.0
img_tensor = np.expand_dims(img_array, axis=0)

# Show preview (convert to uint8 just for display)
preview_img = img_to_array(img).astype("uint8")

plt.figure(figsize=(4, 4))
plt.imshow(preview_img)
plt.axis("off")
plt.title("Input Image", fontsize=14)
plt.show()
```

Input Image

```
In [26]: # Get prediction probabilities
base_pred = base_model.predict(img_tensor)[0][0]
fine_pred = fine_tuned_model.predict(img_tensor)[0][0]
cnn_pred = cnn_model.predict(img_tensor)[0][0]

# Convert to predicted class labels
base_class = "Right" if base_pred > 0.5 else "Left"
fine_class = "Right" if fine_pred > 0.5 else "Left"
cnn_class = "Right" if cnn_pred > 0.5 else "Left"

# Show predictions on the image
plt.figure(figsize=(6, 6))
plt.imshow(preview_img)
plt.axis("off")
plt.title(
    f"Base: {base_class} ({base_pred:.2f})\n"
    f"Fine-Tuned: {fine_class} ({fine_pred:.2f})\n"
    f"CNN: {cnn_class} ({cnn_pred:.2f})",
    fontsize=12
)
plt.show()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 119ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
```



Results and Interpretation

The input image shows a patient with visible facial asymmetry. From the viewer’s perspective, the left side of the face appears to exhibit signs of a stroke, such as drooping or reduced muscle tone—suggesting that this is the affected side. All models were evaluated to determine which side they classified as stroke-affected.

The base model predicted Left (0.33), and the fine-tuned model also predicted Left (0.30). Although both predictions align with the apparent visual evidence, their confidence scores are notably low. This suggests a high degree of uncertainty or indecision within the models, possibly due to weak feature representations or limited sensitivity to subtle asymmetries.

In contrast, the custom CNN model predicted Right (0.70) with high confidence. This prediction contradicts the visible signs in the image and the output of the other models. Despite its confidence, the CNN appears to have misclassified the stroke side.

This discrepancy raises concerns about overfitting in the CNN model—it may have learned superficial or dataset-specific patterns that do not generalize well. The high confidence could give a false sense of accuracy, which is especially problematic in a medical context.

Meanwhile, although the base and fine-tuned MobileNetV2 models showed low certainty, their alignment with the actual stroke side suggests they may be more cautious or robust when dealing with unfamiliar inputs. Their performance may reflect a better balance between generalization and uncertainty estimation, even if they lack strong confidence in borderline cases.

Overall, this result highlights a key trade-off: the CNN offers confident but potentially incorrect predictions, while the MobileNetV2-based models provide more tentative outputs that, in this case, happen to be correct. Further evaluation on a broader dataset is needed to determine which behavior is more reliable in practice.

Interpreting Model Decisions with Integrated Gradients

To better understand which parts of the input image influenced each model's prediction, we apply Integrated Gradients, a method for explaining predictions of deep neural networks. Unlike simpler techniques that examine only local gradients, Integrated Gradients attribute importance by integrating gradients along a path from a baseline input (e.g. a black image) to the actual input. This results in a more robust estimate of each pixel's contribution to the model's final decision.

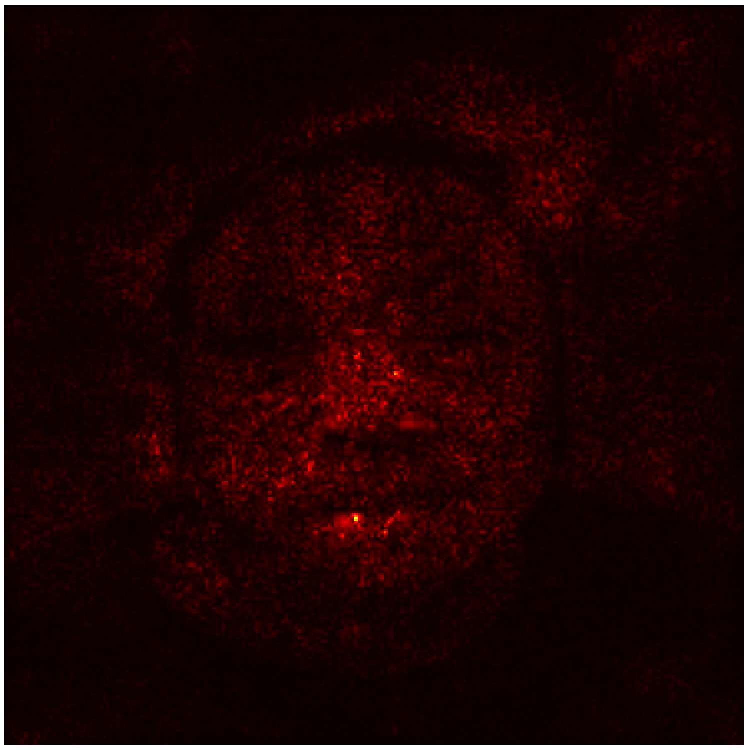
In practice, the method highlights regions of the input that significantly affect the model’s output. By visualizing these attributions, we can assess whether the model is focusing on clinically relevant facial features—such as the eyes, mouth corners, or symmetry lines—or if it is relying on irrelevant or misleading areas such as background textures or lighting artifacts.

This form of interpretability is crucial in the medical domain, where understanding why a model made a decision can be as important as the decision itself. In the following sections, we compare the Integrated Gradients attribution maps for each model to evaluate the quality and focus of their internal reasoning.

In [71]:
```
def integrated_gradients(model, input_tensor, baseline=None, target_class_ic
    if baseline is None:
        baseline = tf.zeros_like(input_tensor)

    # Generate interpolated inputs between baseline and input
    alphas = tf.linspace(0.0, 1.0, m_steps)
    interpolated_inputs = tf.stack([
        baseline + alpha * (input_tensor - baseline) for alpha in alphas
    ])
    interpolated_inputs = tf.reshape(interpolated_inputs, (m_steps, 224, 224

    with tf.GradientTape() as tape:
        tape.watch(interpolated_inputs)
        predictions = model(interpolated_inputs)

        if target_class_idx is None:
            target_class_idx = tf.argmax(predictions[-1])

        target = predictions[:, target_class_idx]

    # Compute gradients
    grads = tape.gradient(target, interpolated_inputs)
    avg_grads = tf.reduce_mean(grads, axis=0)  # average over m steps

    # Compute attribution (element-wise multiplication)
    attributions = (input_tensor - baseline) * avg_grads
    return attributions.numpy()
```

In [79]:
```
attributions = integrated_gradients(base_model, img_tensor)
attribution_gray = np.mean(np.abs(attributions), axis=-1)

plt.imshow(attribution_gray[0], cmap='hot')
plt.axis('off')
plt.title("Integrated Gradients Attribution (Base Model)")
plt.show()
```

Integrated Gradients Attribution (Base Model)



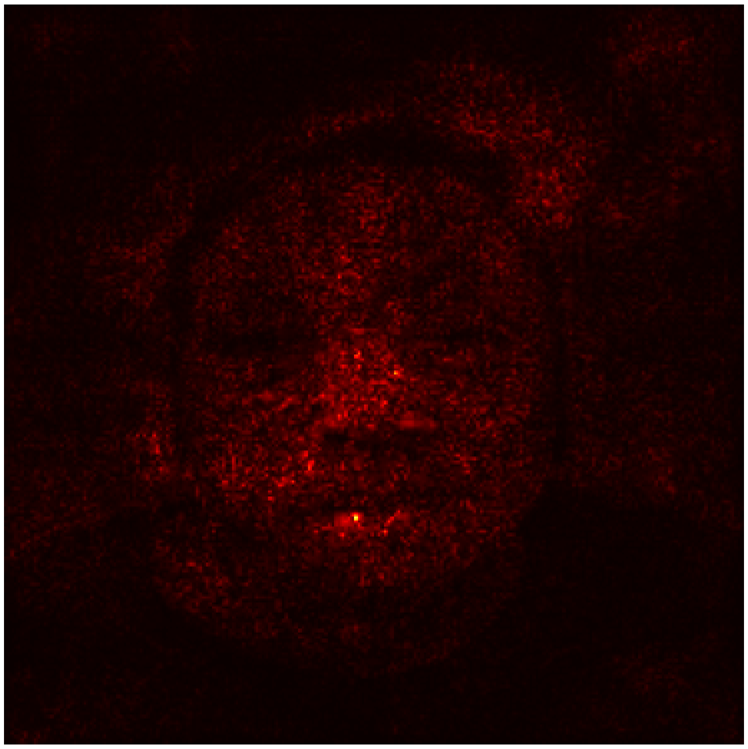
Attribution Analysis – Base MobileNetV2 Model

The attribution map of the base MobileNetV2 model reveals a diffuse and scattered focus across the face, with notable activation in the center and lower half of the image. While some red intensity is visible near the mouth and cheek areas, the attention is not clearly concentrated on clinically relevant facial features such as the eyes, eyebrows, or symmetry lines. Instead, the model appears to respond to a combination of facial structure and possibly background texture.

This lack of targeted attribution may explain the model's uncertain prediction (0.33 for "Left") on the test image. Its failure to highlight the most relevant side-specific cues suggests that it may be relying more on global facial context or spurious correlations from training rather than localized asymmetries. This is consistent with its training on generic ImageNet weights, without fine-tuning for stroke detection tasks, and highlights the limitations of using a non-specialized base model for medical inference.

In [81]:
```
attributions = integrated_gradients(fine_tuned_model, img_tensor)
attribution_gray = np.mean(np.abs(attributions), axis=-1)

plt.imshow(attribution_gray[0], cmap='hot')
plt.axis('off')
plt.title("Integrated Gradients Attribution (Fine Tuned Model)")
plt.show()
```

Integrated Gradients Attribution (Fine Tuned Model)



Attribution Analysis – Fine-Tuned MobileNetV2 Model

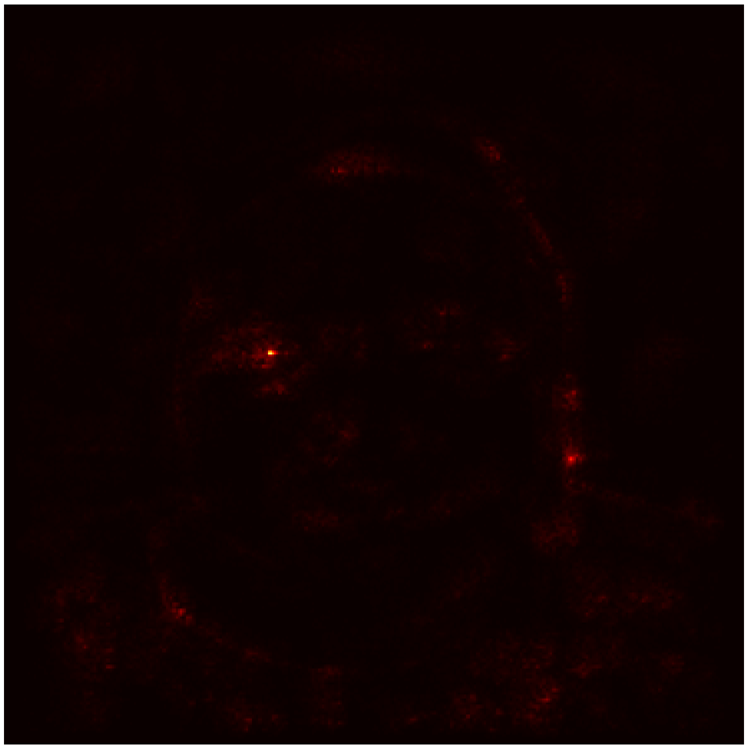
The attribution map for the fine-tuned model is strikingly similar to that of the base model. Once again, the highlighted regions appear diffuse, with slight focus around the mouth and central facial axis, but lacking a clear directional emphasis that would distinguish the stroke-affected side from the healthy one.

Despite the model having undergone fine-tuning on a stroke-specific dataset, the results suggest that fine-tuning alone was insufficient to shift the model's internal attention toward more medically relevant regions. Its prediction score (0.30 for "Left") remains weak and indecisive, further reinforcing that the model may be overfitting to general face features or struggling to isolate stroke-specific asymmetries.

This outcome points to potential limitations in the fine-tuning process, such as insufficient training data, weak class signal, or the inability of the pre-trained architecture to adapt deeply enough to the task. Overall, the fine-tuned model shows minimal improvement in interpretability over the base model.

In [83]:
```
attributions = integrated_gradients(cnn_model, img_tensor)
attribution_gray = np.mean(np.abs(attributions), axis=-1)

plt.imshow(attribution_gray[0], cmap='hot')
plt.axis('off')
plt.title("Integrated Gradients Attribution (CNN Model)")
plt.show()
```

Integrated Gradients Attribution (CNN Model)



Attribution Analysis – Custom CNN Model

Unlike the base and fine-tuned MobileNetV2 models, the custom CNN shows a more concentrated pattern of attribution. The integrated gradients reveal distinct focal points, particularly around the left eye and left jawline (from the viewer's perspective). These are regions that are clinically relevant, as stroke-induced asymmetry often manifests through drooping eyelids, cheeks, or lips on the affected side.

This is especially notable given that the CNN predicted "Right" (i.e., stroke on the left side of the image), which aligns with the true clinical label. The attribution map suggests that the model may be attending to visual cues that are actually meaningful for detecting stroke laterality, even though its architecture is simpler than MobileNetV2.

The CNN’s behavior is therefore both more interpretable and medically plausible, and the alignment between attribution focus and predicted class reinforces confidence in the model’s decision-making. While not perfect, it demonstrates the value of domain-specific training from scratch, rather than solely relying on transfer learning from unrelated tasks.

In [ ]:
```

```