In [1]:
```
# %% 0. Imports & runtime setup
import os, requests, types, numpy as np, pandas as pd, tensorflow as tf
from sklearn.utils import class_weight
from tensorflow.keras import layers, mixed_precision

# Enable mixed precision on any detected GPU (handy for RTX 5090)
if tf.config.list_physical_devices("GPU"):
    mixed_precision.set_global_policy("mixed_float16")

# Simple args object for notebook use
Args = types.SimpleNamespace
args = Args(
    csv="all_countries_videos_cleaned.csv",  # path to your metadata CSV
    img_dir="thumbnails",                    # folder for thumbnails
    model="efficientnetv2l",                 # "resnet50" or "efficientne
    batch=64,
    epochs=5,
    unfreeze=10,                             # fine-tune last N layers
)
print(args)
```

```
2025-06-06 00:54:14.290485: I tensorflow/core/util/port.cc:153] oneDNN cus
tom operations are on. You may see slightly different numerical results du
e to floating-point round-off errors from different computation orders. To
turn them off, set the environment variable `TF_ENABLE_ONEDNN_OPTS=0`.
2025-06-06 00:54:14.305872: E external/local_xla/xla/stream_executor/cuda/
cuda_fft.cc:477] Unable to register cuFFT factory: Attempting to register
factory for plugin cuFFT when one has already been registered
WARNING: All log messages before absl::InitializeLog() is called are writt
en to STDERR
E0000 00:00:1749196454.320765    2339 cuda_dnn.cc:8310] Unable to register
cuDNN factory: Attempting to register factory for plugin cuDNN when one ha
s already been registered
E0000 00:00:1749196454.325070    2339 cuda_blas.cc:1418] Unable to registe
r cuBLAS factory: Attempting to register factory for plugin cuBLAS when on
e has already been registered
2025-06-06 00:54:14.340418: I tensorflow/core/platform/cpu_feature_guard.c
c:210] This TensorFlow binary is optimized to use available CPU instructio
ns in performance-critical operations.
To enable the following instructions: AVX2 AVX_VNNI FMA, in other operatio
ns, rebuild TensorFlow with the appropriate compiler flags.
namespace(csv='all_countries_videos_cleaned.csv', img_dir='thumbnails', mo
del='efficientnetv2l', batch=64, epochs=5, unfreeze=10)
```

In [2]:
```
# %% 1. Backbone registry (swap light vs heavy CNN)
from tensorflow.keras.applications import ResNet50, EfficientNetV2L
from tensorflow.keras.applications.resnet50 import preprocess_input as rn
from tensorflow.keras.applications.efficientnet_v2 import preprocess_inpu

MODELS = {
    "resnet50": {
        "cls": ResNet50,
        "img_size": (224, 224),
        "preprocess": rn_preprocess,
    },
    "efficientnetv2l": {
        "cls": EfficientNetV2L,
        "img_size": (384, 384),
        "preprocess": ev2_preprocess,
    },
}
conf = MODELS[args.model]
IMG_SIZE = conf["img_size"]
```

In [3]:
```
# %% 1. Read CSV and prepare image paths + label
df = pd.read_csv(args.csv)

# Create binary label: 1 if viral, 0 otherwise
df["is_viral"] = ((df["views"] > 100_000) & (df["likes"] > 10_000)).astyp

# Construct local image path using video_id (e.g., thumbnails/abc123.jpg)
df["thumbnail_path"] = df["video_id"].apply(lambda vid: f"{args.img_dir}/

# Remove rows where the image file doesn't exist
df = df[df["thumbnail_path"].apply(os.path.exists)].reset_index(drop=True

print("✅ Valid samples:", len(df))
```

```
✅ Valid samples: 254606
```

In [4]:
```
# %% 3. tf.data pipeline (image branch + TF-IDF title)
AUTOTUNE = tf.data.AUTOTUNE
VOCAB = 2000

# Build TextVectorization layer using TF-IDF
vectorize = layers.TextVectorization(
    max_tokens=VOCAB,
    output_mode="tf_idf"
)
vectorize.adapt(df["title"])

# Image loading and preprocessing
def load_image(path):
    img = tf.io.read_file(path)
    img = tf.image.decode_jpeg(img, channels=3)
    img = tf.image.resize(img, IMG_SIZE)
    img = conf["preprocess"](img)
    return img

# Wrap inputs into a dictionary
def pack(title, img_path, label):
    return {"title": title, "thumbnail": img_path}, label

# Apply preprocessing to both title and image
def preprocess(inputs, label):
    title = vectorize(inputs["title"])
    thumbnail = load_image(inputs["thumbnail"])
    label = tf.cast(label, tf.float32)
    label = tf.expand_dims(label, -1)
    return {"title": title, "thumbnail": thumbnail}, label

# Build the full dataset
ds = (
    tf.data.Dataset.from_tensor_slices((df["title"].values, df["thumbnail
    .map(pack, num_parallel_calls=AUTOTUNE)
    .map(preprocess, num_parallel_calls=AUTOTUNE)
    .shuffle(2048)
    .batch(args.batch)
    .prefetch(AUTOTUNE)
)
```

```
I0000 00:00:1749196827.731497    2339 gpu_device.cc:2022] Created device /
job:localhost/replica:0/task:0/device:GPU:0 with 13499 MB memory:  -> devi
ce: 0, name: NVIDIA GeForce RTX 4080 SUPER, pci bus id: 0000:01:00.0, comp
ute capability: 8.9
```

In [5]:

```
# %% 4. Build multimodal model
# Text branch
text_in = tf.keras.Input(shape=(VOCAB,), name="title")
text_x = layers.Dense(256, activation="relu")(text_in)
text_x = layers.Dropout(0.3)(text_x)

# Image branch
img_in = tf.keras.Input(shape=(*IMG_SIZE, 3), name="thumbnail")
backbone = conf["cls"](include_top=False, weights="imagenet", pooling="av
backbone.trainable = False     # phase 1 frozen
img_x = backbone(img_in, training=False)

# Fusion & output
fusion = layers.concatenate([text_x, img_x])
fusion = layers.Dense(128, activation="relu")(fusion)
fusion = layers.Dropout(0.3)(fusion)
out = layers.Dense(1, activation="sigmoid", dtype="float32")(fusion)

model = tf.keras.Model(inputs={"title": text_in, "thumbnail": img_in}, ou
model.summary()
```

Model: "functional"

```
┏━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━┳━━━━━━━━━━━━━
┃ Layer (type)         ┃ Output Shape       ┃     Param # ┃ Connected t
┡━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━╇━━━━━━━━━━━━━
│ title (InputLayer)   │ (None, 2000)       │           0 │ -
├──────────────────────┼────────────────────┼─────────────┼─────────────
│ cast (Cast)          │ (None, 2000)       │           0 │ title[0][0
├──────────────────────┼────────────────────┼─────────────┼─────────────
│ dense (Dense)        │ (None, 256)        │     512,256 │ cast[0][0]
├──────────────────────┼────────────────────┼─────────────┼─────────────
│ thumbnail            │ (None, 384, 384,   │           0 │ -
│ (InputLayer)         │ 3)                 │             │
├──────────────────────┼────────────────────┼─────────────┼─────────────
│ dropout (Dropout)    │ (None, 256)        │           0 │ dense[0][0
├──────────────────────┼────────────────────┼─────────────┼─────────────
│ efficientnetv2-l     │ (None, 1280)       │ 117,746,8… │ thumbnail[
│ (Functional)         │                    │             │
├──────────────────────┼────────────────────┼─────────────┼─────────────
│ concatenate          │ (None, 1536)       │           0 │ dropout[0]
│ (Concatenate)        │                    │             │ efficientne
├──────────────────────┼────────────────────┼─────────────┼─────────────
│ dense_1 (Dense)      │ (None, 128)        │     196,736 │ concatenate
├──────────────────────┼────────────────────┼─────────────┼─────────────
│ dropout_1 (Dropout)  │ (None, 128)        │           0 │ dense_1[0]
├──────────────────────┼────────────────────┼─────────────┼─────────────
│ cast_2 (Cast)        │ (None, 128)        │           0 │ dropout_1[
├──────────────────────┼────────────────────┼─────────────┼─────────────
│ dense_2 (Dense)      │ (None, 1)          │         129 │ cast_2[0][
└──────────────────────┴────────────────────┴─────────────┴─────────────
 Total params: 118,455,969 (451.87 MB)
 Trainable params: 709,121 (2.71 MB)
 Non-trainable params: 117,746,848 (449.17 MB)
```

In [6]:

```python
# %% 5. Phase 1 training (only top layers)
labels_np = df.is_viral.values
cw = class_weight.compute_class_weight("balanced", classes=np.unique(labe
class_weights = {i: w for i, w in enumerate(cw)}

model.compile(optimizer="adam", loss="binary_crossentropy", metrics=["acc
history1 = model.fit(ds, epochs=args.epochs, class_weight=class_weights)
```

```
Epoch 1/5
```

```
WARNING: All log messages before absl::InitializeLog() is called are writt
en to STDERR
I0000 00:00:1749197058.668381    2586 service.cc:148] XLA service 0x7f40a8
dc2f90 initialized for platform CUDA (this does not guarantee that XLA wil
l be used). Devices:
I0000 00:00:1749197058.668942    2586 service.cc:156]   StreamExecutor dev
ice (0): NVIDIA GeForce RTX 4080 SUPER, Compute Capability 8.9
2025-06-06 01:04:20.357010: I tensorflow/compiler/mlir/tensorflow/utils/du
mp_mlir_util.cc:268] disabling MLIR crash reproducer, set env var `MLIR_CR
ASH_REPRODUCER_DIRECTORY` to enable.
I0000 00:00:1749197063.915557    2586 cuda_dnn.cc:529] Loaded cuDNN versio
n 90300
2025-06-06 01:04:33.789684: E external/local_xla/xla/service/slow_operatio
n_alarm.cc:65] Trying algorithm eng0{} for conv (f16[64,96,96,256]{3,2,1,
0}, u8[0]{0}) custom-call(f16[64,96,96,64]{3,2,1,0}, f16[256,3,3,64]{3,2,
1,0}), window={size=3x3 pad=1_1x1_1}, dim_labels=b01f_o01i->b01f, custom_c
all_target="__cudnn$convForward", backend_config={"cudnn_conv_backend_conf
ig":{"activation_mode":"kNone","conv_result_scale":1,"leakyrelu_alpha":0,"
side_input_scale":0},"force_earliest_schedule":false,"operation_queue_i
d":"0","wait_on_operation_queues":[]} is taking a while...
2025-06-06 01:04:34.101704: E external/local_xla/xla/service/slow_operatio
n_alarm.cc:133] The operation took 1.314447356s
Trying algorithm eng0{} for conv (f16[64,96,96,256]{3,2,1,0}, u8[0]{0}) cu
stom-call(f16[64,96,96,64]{3,2,1,0}, f16[256,3,3,64]{3,2,1,0}), window={si
ze=3x3 pad=1_1x1_1}, dim_labels=b01f_o01i->b01f, custom_call_target="__cud
nn$convForward", backend_config={"cudnn_conv_backend_config":{"activation_
mode":"kNone","conv_result_scale":1,"leakyrelu_alpha":0,"side_input_scal
e":0},"force_earliest_schedule":false,"operation_queue_id":"0","wait_on_op
eration_queues":[]} is taking a while...
I0000 00:00:1749197090.877400    2586 device_compiler.h:188] Compiled clus
ter using XLA!  This line is logged at most once for the lifetime of the p
rocess.
3979/3979 ━━━━━━━━━━━━━━━━━━━━ 609s 140ms/step - accuracy: 0.8053 - loss:
0.4147
Epoch 2/5
2025-06-06 01:14:08.533122: W tensorflow/core/kernels/data/prefetch_autotu
ner.cc:52] Prefetch autotuner tried to allocate 113758464 bytes after enco
untering the first element of size 113758464 bytes.This already causes the
autotune ram budget to be exceeded. To stay within the ram budget, either
increase the ram budget or reduce element size
2025-06-06 01:14:08.534033: W tensorflow/core/kernels/data/prefetch_autotu
ner.cc:52] Prefetch autotuner tried to allocate 113758976 bytes after enco
untering the first element of size 113758976 bytes.This already causes the
autotune ram budget to be exceeded. To stay within the ram budget, either
increase the ram budget or reduce element size
3979/3979 ━━━━━━━━━━━━━━━━━━━━ 660s 166ms/step - accuracy: 0.8502 - loss:
0.3361
Epoch 3/5
   1/3979 ━━━━━━━━━━━━━━━━━━━━ 42:50 646ms/step - accuracy: 0.8281 - loss:
0.3258
```

```
2025-06-06 01:25:08.455611: W tensorflow/core/kernels/data/prefetch_autotu
ner.cc:52] Prefetch autotuner tried to allocate 113758464 bytes after enco
untering the first element of size 113758464 bytes.This already causes the
autotune ram budget to be exceeded. To stay within the ram budget, either
increase the ram budget or reduce element size
2025-06-06 01:25:08.455864: W tensorflow/core/kernels/data/prefetch_autotu
ner.cc:52] Prefetch autotuner tried to allocate 113758976 bytes after enco
untering the first element of size 113758976 bytes.This already causes the
autotune ram budget to be exceeded. To stay within the ram budget, either
increase the ram budget or reduce element size
3979/3979 ━━━━━━━━━━━━━━━━━━━━ 532s 134ms/step - accuracy: 0.8781 - loss:
0.2822
Epoch 4/5
   1/3979 ━━━━━━━━━━━━━━━━━━━━ 41:24 625ms/step - accuracy: 0.8281 - loss:
0.3756
2025-06-06 01:34:00.546095: W tensorflow/core/kernels/data/prefetch_autotu
ner.cc:52] Prefetch autotuner tried to allocate 113758464 bytes after enco
untering the first element of size 113758464 bytes.This already causes the
autotune ram budget to be exceeded. To stay within the ram budget, either
increase the ram budget or reduce element size
2025-06-06 01:34:00.546302: W tensorflow/core/kernels/data/prefetch_autotu
ner.cc:52] Prefetch autotuner tried to allocate 113758976 bytes after enco
untering the first element of size 113758976 bytes.This already causes the
autotune ram budget to be exceeded. To stay within the ram budget, either
increase the ram budget or reduce element size
3979/3979 ━━━━━━━━━━━━━━━━━━━━ 531s 133ms/step - accuracy: 0.8943 - loss:
0.2484
Epoch 5/5
   1/3979 ━━━━━━━━━━━━━━━━━━━━ 43:02 649ms/step - accuracy: 0.9219 - loss:
0.2106
2025-06-06 01:42:51.289630: W tensorflow/core/kernels/data/prefetch_autotu
ner.cc:52] Prefetch autotuner tried to allocate 113758464 bytes after enco
untering the first element of size 113758464 bytes.This already causes the
autotune ram budget to be exceeded. To stay within the ram budget, either
increase the ram budget or reduce element size
2025-06-06 01:42:51.289889: W tensorflow/core/kernels/data/prefetch_autotu
ner.cc:52] Prefetch autotuner tried to allocate 113758976 bytes after enco
untering the first element of size 113758976 bytes.This already causes the
autotune ram budget to be exceeded. To stay within the ram budget, either
increase the ram budget or reduce element size
3979/3979 ━━━━━━━━━━━━━━━━━━━━ 530s 133ms/step - accuracy: 0.9058 - loss:
0.2231
```

In [7]:
```python
# %% 6. Phase 2 fine-tuning & save
for layer in backbone.layers[-args.unfreeze:]:
    layer.trainable = True

model.compile(optimizer=tf.keras.optimizers.Adam(1e-5),
              loss="binary_crossentropy", metrics=["accuracy"])
history2 = model.fit(ds, epochs=args.epochs, class_weight=class_weights)

model.save("viral_predictor.h5")
print("✓ Training complete – model saved to viral_predictor.h5")
```

```
Epoch 1/5
2025-06-06 01:52:10.800419: W tensorflow/core/kernels/data/prefetch_autotu
ner.cc:52] Prefetch autotuner tried to allocate 113758976 bytes after enco
untering the first element of size 113758976 bytes.This already causes the
autotune ram budget to be exceeded. To stay within the ram budget, either
increase the ram budget or reduce element size
3979/3979 ━━━━━━━━━━━━━━━━━━━━ 623s 144ms/step - accuracy: 0.8791 - loss:
0.2996
Epoch 2/5
   1/3979 ━━━━━━━━━━━━━━━━━━━━ 43:43 660ms/step - accuracy: 0.9375 - loss:
0.1749
2025-06-06 02:02:08.013509: W tensorflow/core/kernels/data/prefetch_autotu
ner.cc:52] Prefetch autotuner tried to allocate 113758464 bytes after enco
untering the first element of size 113758464 bytes.This already causes the
autotune ram budget to be exceeded. To stay within the ram budget, either
increase the ram budget or reduce element size
2025-06-06 02:02:08.013780: W tensorflow/core/kernels/data/prefetch_autotu
ner.cc:52] Prefetch autotuner tried to allocate 113758976 bytes after enco
untering the first element of size 113758976 bytes.This already causes the
autotune ram budget to be exceeded. To stay within the ram budget, either
increase the ram budget or reduce element size
3979/3979 ━━━━━━━━━━━━━━━━━━━━ 544s 137ms/step - accuracy: 0.8970 - loss:
0.2422
Epoch 3/5
   1/3979 ━━━━━━━━━━━━━━━━━━━━ 42:51 646ms/step - accuracy: 0.8125 - loss:
0.3830
2025-06-06 02:11:11.686930: W tensorflow/core/kernels/data/prefetch_autotu
ner.cc:52] Prefetch autotuner tried to allocate 113758464 bytes after enco
untering the first element of size 113758464 bytes.This already causes the
autotune ram budget to be exceeded. To stay within the ram budget, either
increase the ram budget or reduce element size
2025-06-06 02:11:11.687113: W tensorflow/core/kernels/data/prefetch_autotu
ner.cc:52] Prefetch autotuner tried to allocate 113758976 bytes after enco
untering the first element of size 113758976 bytes.This already causes the
autotune ram budget to be exceeded. To stay within the ram budget, either
increase the ram budget or reduce element size
3979/3979 ━━━━━━━━━━━━━━━━━━━━ 536s 135ms/step - accuracy: 0.9032 - loss:
0.2275
Epoch 4/5
   1/3979 ━━━━━━━━━━━━━━━━━━━━ 41:13 622ms/step - accuracy: 0.8750 - loss:
0.3901
2025-06-06 02:20:07.909571: W tensorflow/core/kernels/data/prefetch_autotu
ner.cc:52] Prefetch autotuner tried to allocate 113758464 bytes after enco
untering the first element of size 113758464 bytes.This already causes the
autotune ram budget to be exceeded. To stay within the ram budget, either
increase the ram budget or reduce element size
2025-06-06 02:20:07.909840: W tensorflow/core/kernels/data/prefetch_autotu
ner.cc:52] Prefetch autotuner tried to allocate 113758976 bytes after enco
untering the first element of size 113758976 bytes.This already causes the
autotune ram budget to be exceeded. To stay within the ram budget, either
increase the ram budget or reduce element size
```

```
3979/3979 ━━━━━━━━━━━━━━━━━━━━ 532s 133ms/step - accuracy: 0.9080 - loss:
0.2186
Epoch 5/5
   1/3979 ━━━━━━━━━━━━━━━━━━━━ 42:20 639ms/step - accuracy: 0.8438 - loss:
0.4214
2025-06-06 02:28:59.486187: W tensorflow/core/kernels/data/prefetch_autotu
ner.cc:52] Prefetch autotuner tried to allocate 113758464 bytes after enco
untering the first element of size 113758464 bytes.This already causes the
autotune ram budget to be exceeded. To stay within the ram budget, either
increase the ram budget or reduce element size
2025-06-06 02:28:59.486450: W tensorflow/core/kernels/data/prefetch_autotu
ner.cc:52] Prefetch autotuner tried to allocate 113758976 bytes after enco
untering the first element of size 113758976 bytes.This already causes the
autotune ram budget to be exceeded. To stay within the ram budget, either
increase the ram budget or reduce element size
3979/3979 ━━━━━━━━━━━━━━━━━━━━ 535s 134ms/step - accuracy: 0.9109 - loss:
0.2125
WARNING:absl:You are saving your model as an HDF5 file via `model.save()`
or `keras.saving.save_model(model)`. This file format is considered legac
y. We recommend using instead the native Keras format, e.g. `model.save('m
y_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
✓ Training complete — model saved to viral_predictor.h5
```