دانشگاه اصفهان

دانشکده مهندسی کامپیوتر – گروه هوش مصنوعی و رباتیک

گزارش تمرین اول – شبکه‌های عصبی

NN

پدیدآورنده:

محمد امین کیانی

۴۰۴۳۶۴۴۰۰۸

دانشجوی کارشناسی ارشد، دانشکده‌ی کامپیوتر، دانشگاه اصفهان، اصفهان،
Aminkianiworkeng@gmail.com

استاد درس: دکتر برادران

نیمسال اول تحصیلی ۰۵-۱۴۰۴

فهرست مطالب

مستندات

۱-مسئله و تحلیل کلی آن:

- بارگذاری دیتاست beans از HuggingFace
- تغییر سایز به ۲۲۴×۲۲۴ و نرمال‌سازی با آمار ImageNet و نمایش قبل/بعد
- شمارش تعداد تصاویر هر کلاس
- تعریف MLP سه‌لایه طبق فرمول صورت سؤال و محاسبه ابعاد«؟»
- حلقه آموزش کامل با TensorBoard و ذخیره بهترین مدل در pickle
- اجرای ۵ آزمایش خواسته‌شده با تنظیمات متفاوت (dropout، نرمال‌سازی، برچسب تصادفی، نویز گاوسی و روش پیشنهادی برای بهبود بدون افزودن نویز به داده‌ی آموزش).

حال، بر اساس این چارچوب:

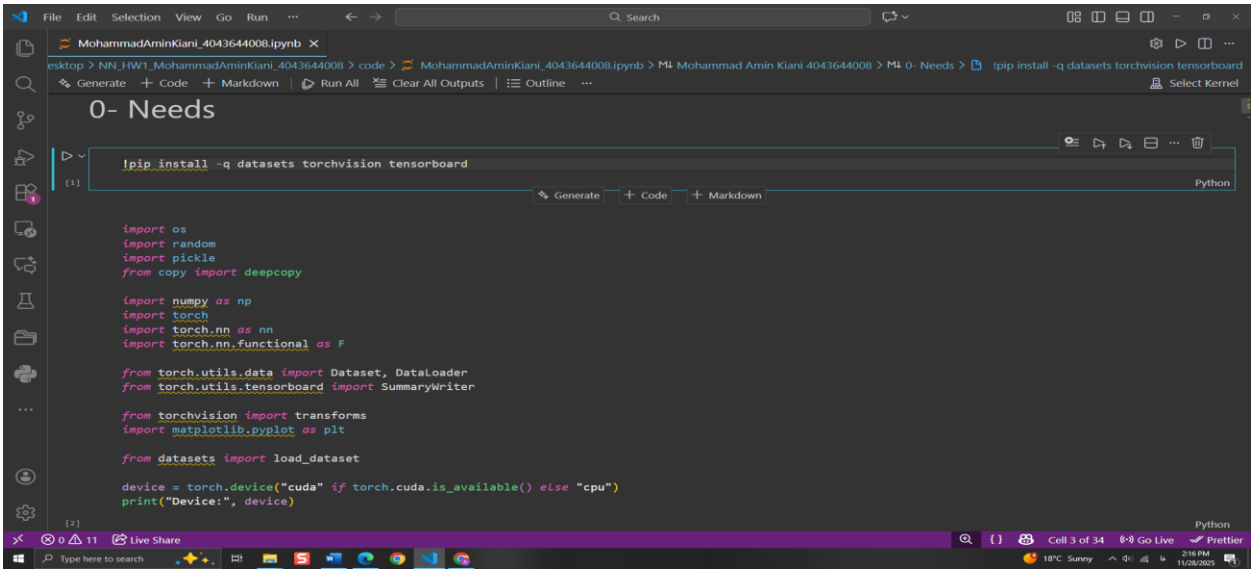
هدف تمرین، طراحی و پیاده‌سازی یک سامانه‌ی یادگیری عمیق برای **تشخیص وضعیت برگ‌های لوبیا** با استفاده از یک **شبکه‌ی عصبی ژرف (MLP سه‌لایه)** است.

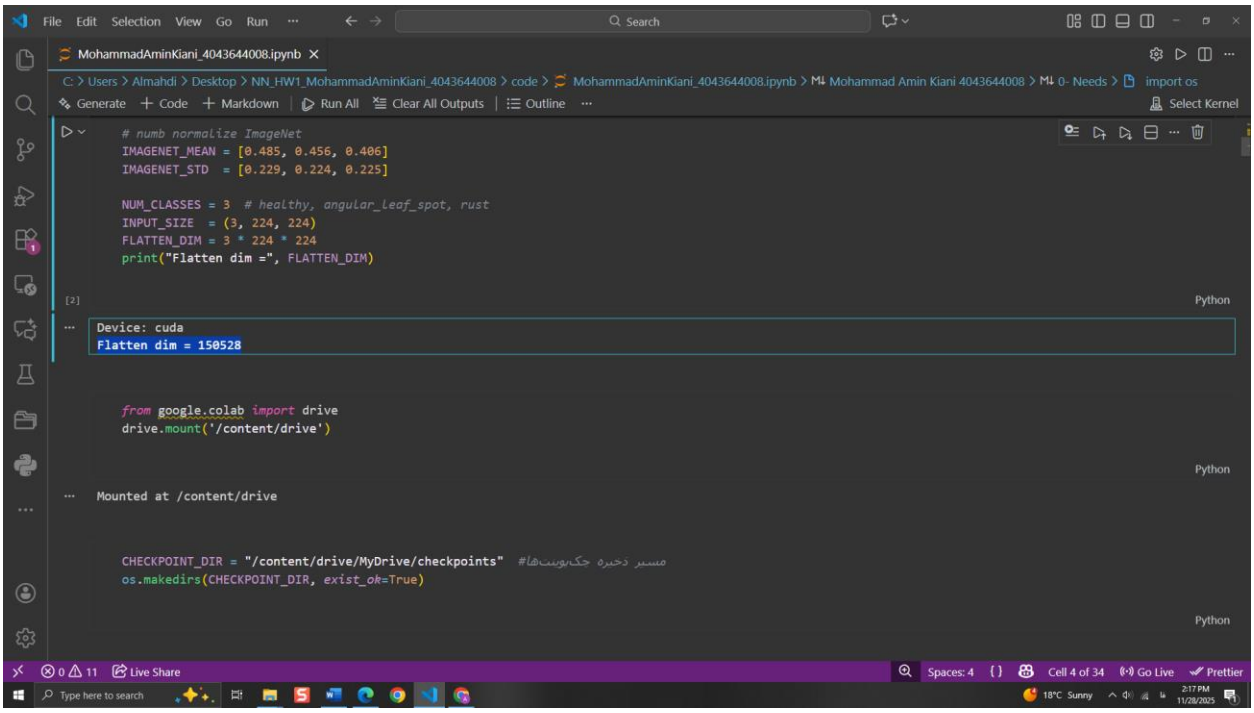
سه کلاس داریم:

- **۰ = لکه‌ی زاویه‌ای برگ(angular_leaf_spot)**
- **۱ = قارچ زنگار(bean_rust)**
- **۲ = برگ سالم(healthy)**

مدل باید با استفاده از تصاویر برگ‌ها، بتواند این سه وضعیت را تشخیص دهد. علاوه بر خود مدل، روی **درک نقش پیش‌پردازش، نرمال‌سازی، dropout، نویز، برچسب اشتباه و روش‌های بهبود پایداری مدل** تمرکز کند.

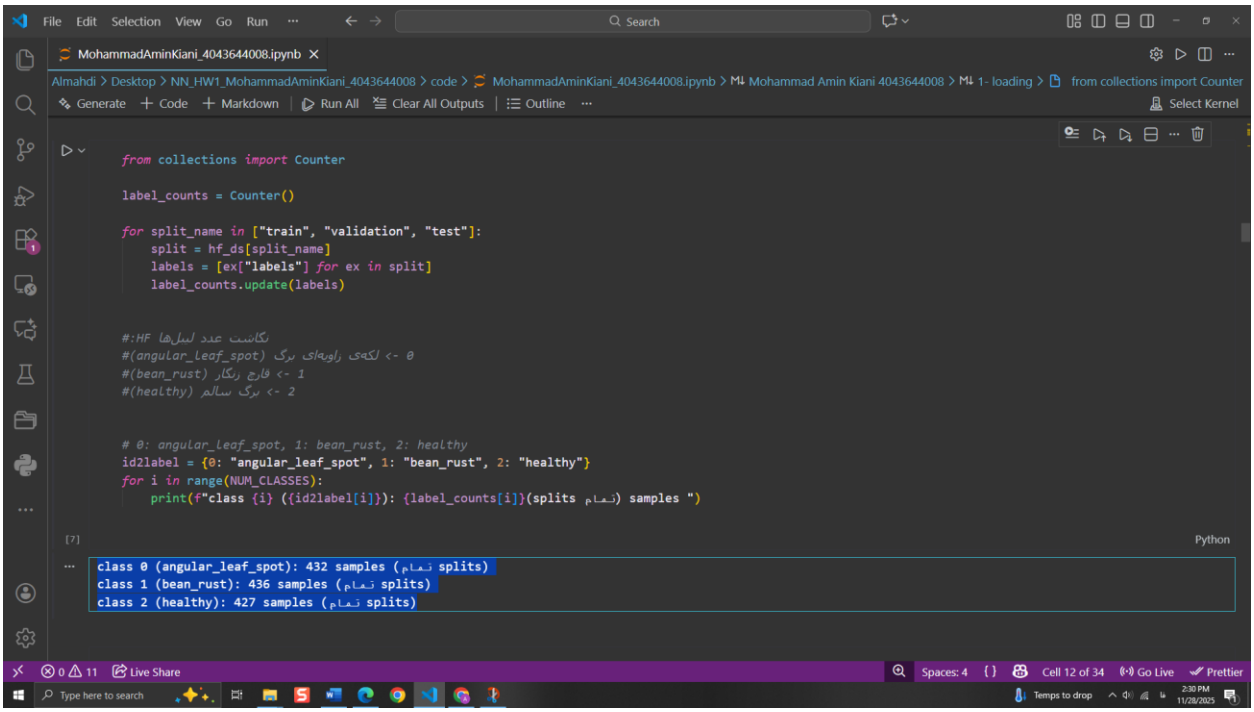
۲- بخش ۱: مجموعه‌داده

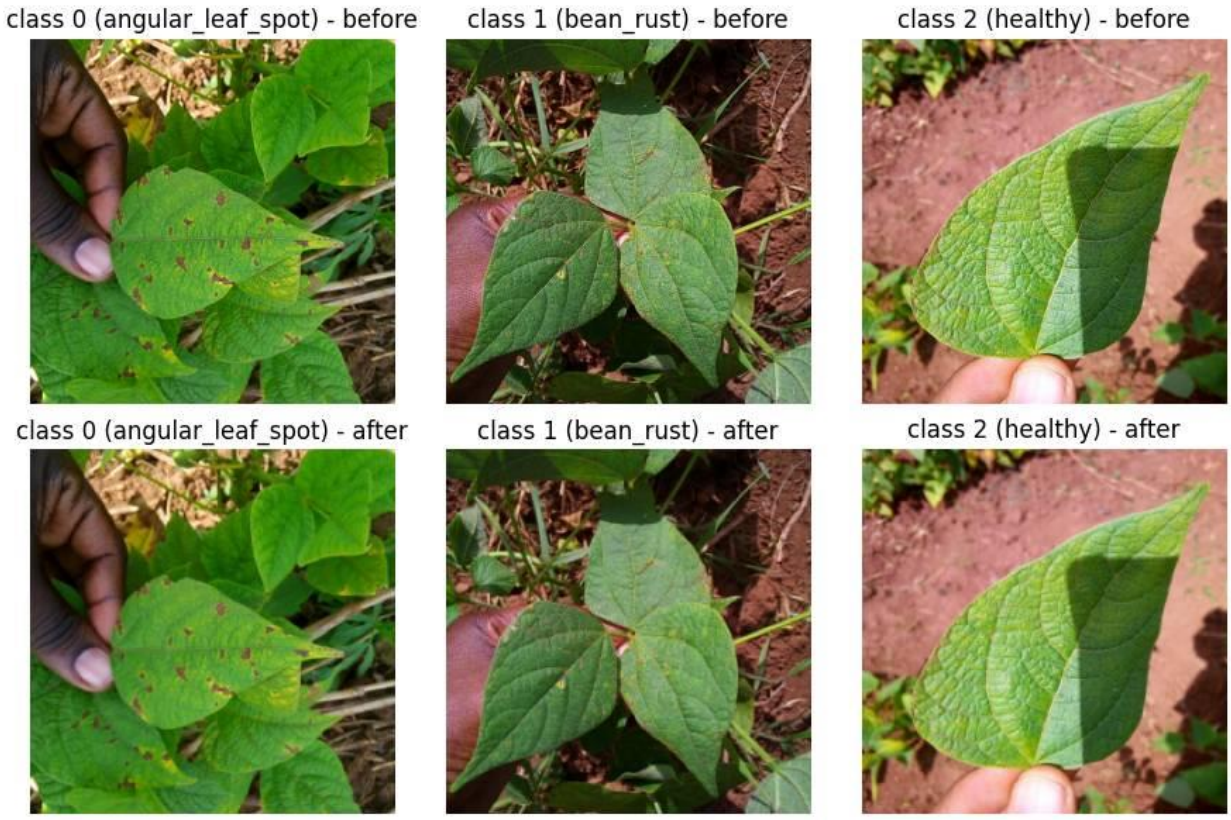




```python
#  train / validation / test
hf_ds = load_dataset("AI-Lab-Makerere/beans")
print(hf_ds)
print(hf_ds["train"][0])
```

```
/usr/local/lib/python3.12/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your Google Colab and restart your sessio
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
README.md: 4.95k/? [00:00<00:00, 441kB/s]
data/train-00000-of-00001.parquet: 100% 144M/144M [00:01<00:00, 220MB/s]
data/validation-00000-of-00001.parquet: 100% 18.5M/18.5M [00:00<00:00, 387kB/s]
data/test-00000-of-00001.parquet: 100% 17.7M/17.7M [00:00<00:00, 52.9MB/s]
Generating train split: 100% 1034/1034 [00:01<00:00, 860.32 examples/s]
Generating validation split: 100% 133/133 [00:00<00:00, 659.53 examples/s]
Generating test split: 100% 128/128 [00:00<00:00, 660.89 examples/s]
DatasetDict({
    train: Dataset({
        features: ['image_file_path', 'image', 'labels'],
        num_rows: 1034
    })
    validation: Dataset({
        features: ['image_file_path', 'image', 'labels'],
        num_rows: 133
    })
    test: Dataset({
        features: ['image_file_path', 'image', 'labels'],
        num_rows: 128
    })
})
{'image_file_path': '/home/albert/.cache/huggingface/datasets/downloads/extracted/967f0d9f61a7a8de58892c6fab6f02317c06faf3e19fba6a07b0885a9a7142c7/train/angular_leaf_spot/angular_lea
```





۳- بخش ۲: طراحی مدل

```python
class BeanMLP(nn.Module):
    def __init__(self, input_dim=FLATTEN_DIM,
                 hidden1=512, hidden2=256,
                 num_classes=NUM_CLASSES,
                 p1=0.3, p2=0.3):
        super().__init__()
        self.flatten = nn.Flatten()
        self.fc1 = nn.Linear(input_dim, hidden1)    # W1: 512 × 150528
        self.fc2 = nn.Linear(hidden1, hidden2)      # W2: 256 × 512
        self.fc3 = nn.Linear(hidden2, num_classes) # W3: 3 × 256
        self.dropout1 = nn.Dropout(p=p1)
        self.dropout2 = nn.Dropout(p=p2)

    def forward(self, x):
        # x: (B, 3, 224, 224)
        x = self.flatten(x)
        o1 = F.relu(self.fc1(x))
        o1 = self.dropout1(o1)
```

```
        o2 = F.relu(self.fc2(o1))
        o2 = self.dropout2(o2)
        logits = self.fc3(o2)  # softmax را به CrossEntropyLoss
        return logits
```

ساختار اصلی:

۱. ورودی:

- تصویر ۳×۲۲۴×۲۲۴
- ابتدا به‌صورت یک بردار تخت (flatten) با طول ۳×۲۲۴×۲۲۴ = ۱۵۰۵۲۸ تبدیل می‌شود.

۲. لایه‌ی مخفی اول:

- وزن‌ها : $W1 \in R^{512 \times 150528}$
- خروجی: بردار ۵۱۲ بعدی
- تابع فعال‌سازی : ReLU
- Dropout با احتمال حذف p۱ (در حالت پایه ۰٫۳)

۳. لایه‌ی مخفی دوم:

- وزن‌ها : $W2 \in R^{256 \times 512}$
- خروجی: بردار ۲۵۶ بعدی
- تابع فعال‌سازی : ReLU
- Dropout با احتمال حذف p۲ (در حالت پایه ۰٫۳)

٤. لایه‌ی خروجی:

- وزن‌ها : $W3 \in R^{3 \times 256}$

- خروجی نهایی: بردار ۳بعدی (logits برای سه کلاس)
- تابع softmax به‌طور ضمنی توسط CrossEntropyLoss اعمال می‌شود.

به‌صورت خلاصه، forward مدل:

- x_flat = Flatten(x)
- o۱ = Dropout(ReLU(W۱ x_flat))
- o۲ = Dropout(ReLU(W۲ o۱))
- logits = W۳ o۲

$$o_1 = dropout(ReLU(W_1 x), p_1) \ \ where \ \ W_1 \in R^{512\times ?}$$
$$o_2 = dropout(ReLU(W_2 o_1), p_2) \ \ where \ \ W_2 \in R^{256\times 512}$$
$$o = softmax(W_3 o_2) \ \ where \ \ W_3 \in R^{?\times 256}$$

- استفاده از num_workers و pin_memory برای بهبود سرعت (در Colab)

۴- بخش ۳: آزمایش‌ها

جدول ۱: ابرپارامترهای مدل

ابرپارامتر	مقدار
سایز تصویر ورودی	$224 \times 224 \times 3$
سایز تصویر اصلی	$500 \times 500 \times 3$
بعد ورودی شبکه	؟
بعد نهان لایه‌ی اول شبکه	512
بعد نهان لایه‌ی دوم شبکه	256
تعداد دسته‌ها	3
تابع فعال‌سازی	ReLU
نرخ حذف تصادفی لایه اول	0.3
نرخ حذف تصادفی لایه دوم	0.3
بهنه‌ساز	Adam
نرخ یادگیری	0.001
size Batch	32
تعداد ایپاک‌ها	15

- علاوه بر وزن‌ها، ما در pickle اطلاعات دیگری هم ذخیره کرده‌ایم:

config آزمایش، شماره ران، شماره ایپاک بهترین مدل، تاریخچه‌ی loss/accuracy و نگاشت id۲label که همگی مرتبط با همان آزمایش‌اند.

- آزمایش اول

```
BASE_CONFIG = {
    "use_normalization": True,
    "hidden1": 512,
    "hidden2": 256,
    "dropout_p1": 0.3,
    "dropout_p2": 0.3,
    "batch_size": 32,
    "num_epochs": 15,
    "lr": 1e-3,
    "random_labels_for_train_val": False,
    "add_gaussian_noise_to_test": False,
    "noise_std": 1.0,
    "seed_base": 42,
}

NUM_RUNS = 20
```

در این آزمایش همه‌ی موارد را نظیر آن چه تعریف شده است ثابت در نظر می‌گیریم:

```
exp1_config = deepcopy(BASE_CONFIG)

stats_exp1, best_exp1 = run_experiment("exp1_base", exp1_config,
num_runs=NUM_RUNS)
```

```
[exp1_base | run 0 | epoch 1/15] train_loss=8.0106, val_loss=3.0776, train_acc=0.510, val_acc=0.609, test_acc=0.516
[exp1_base | run 0 | epoch 2/15] train_loss=2.9738, val_loss=1.5194, train_acc=0.603, val_acc=0.639, test_acc=0.523
[exp1_base | run 0 | epoch 3/15] train_loss=1.6423, val_loss=1.2438, train_acc=0.668, val_acc=0.647, test_acc=0.633
[exp1_base | run 0 | epoch 4/15] train_loss=0.9077, val_loss=0.8593, train_acc=0.719, val_acc=0.669, test_acc=0.719
[exp1_base | run 0 | epoch 5/15] train_loss=0.6895, val_loss=0.8051, train_acc=0.766, val_acc=0.669, test_acc=0.703
[exp1_base | run 0 | epoch 6/15] train_loss=0.5123, val_loss=0.7612, train_acc=0.809, val_acc=0.677, test_acc=0.703
[exp1_base | run 0 | epoch 7/15] train_loss=0.3966, val_loss=0.7511, train_acc=0.824, val_acc=0.699, test_acc=0.742
[exp1_base | run 0 | epoch 8/15] train_loss=0.4260, val_loss=1.1088, train_acc=0.857, val_acc=0.669, test_acc=0.758
[exp1_base | run 0 | epoch 9/15] train_loss=0.3482, val_loss=0.9367, train_acc=0.890, val_acc=0.714, test_acc=0.727
[exp1_base | run 0 | epoch 10/15] train_loss=0.2827, val_loss=0.9259, train_acc=0.903, val_acc=0.707, test_acc=0.711
[exp1_base | run 0 | epoch 11/15] train_loss=0.2888, val_loss=0.8628, train_acc=0.898, val_acc=0.729, test_acc=0.734
[exp1_base | run 0 | epoch 12/15] train_loss=0.1973, val_loss=0.9912, train_acc=0.929, val_acc=0.744, test_acc=0.719
[exp1_base | run 0 | epoch 13/15] train_loss=0.3570, val_loss=1.2900, train_acc=0.881, val_acc=0.662, test_acc=0.688
[exp1_base | run 0 | epoch 14/15] train_loss=0.3053, val_loss=0.9532, train_acc=0.888, val_acc=0.707, test_acc=0.773
[exp1_base | run 0 | epoch 15/15] train_loss=0.2799, val_loss=1.0312, train_acc=0.918, val_acc=0.692, test_acc=0.711
================================================================================
شروع آزمایش exp1_base - 1 ران
[exp1_base | run 1 | epoch 1/15] train_loss=8.5838, val_loss=4.9616, train_acc=0.499, val_acc=0.617, test_acc=0.578
[exp1_base | run 1 | epoch 2/15] train_loss=3.3434, val_loss=1.7923, train_acc=0.616, val_acc=0.624, test_acc=0.562
[exp1_base | run 1 | epoch 3/15] train_loss=1.5840, val_loss=1.2779, train_acc=0.647, val_acc=0.647, test_acc=0.711
[exp1_base | run 1 | epoch 4/15] train_loss=0.9691, val_loss=0.9032, train_acc=0.691, val_acc=0.639, test_acc=0.656
[exp1_base | run 1 | epoch 5/15] train_loss=0.6514, val_loss=1.1424, train_acc=0.769, val_acc=0.699, test_acc=0.742
[exp1_base | run 1 | epoch 6/15] train_loss=0.5863, val_loss=1.0081, train_acc=0.790, val_acc=0.692, test_acc=0.734
...
Test  loss mean±std: 0.7898 ± 0.0479
Test  acc  mean±std: 0.6988 ± 0.0358
بهترین ران: 14 با val_loss = 0.6837
مدل و سایر اطلاعات در 'saved_models/best_model_exp1_base.pkl' ذخیره شد.
```

پس مدل باید به دقت نسبتاً خوبی روی train/val/test برسد (با توجه به سادگی MLP و رزولوشن بالا، بسته به seed). هر ۲۰ اجرا به‌خاطر تفاوت در initialization و شافل، کمی تفاوت در دقت خواهند داشت.

- آزمایش دوم

```
exp2_config = deepcopy(BASE_CONFIG)
exp2_config["dropout_p1"] = 0.6
exp2_config["dropout_p2"] = 0.6

stats_exp2, best_exp2 = run_experiment("exp2_high_dropout", exp2_config,
num_runs=NUM_RUNS)
```

در این آزمایش نرخ حذف تصادفی را به ۰.۶ افزایش می‌دهیم.

```
[exp2_high_dropout | run 0 | epoch 1/15] train_loss=17.2185, val_loss=4.9835, train_acc=0.440, val_acc=0.594, test_acc=0.578
[exp2_high_dropout | run 0 | epoch 2/15] train_loss=8.8425, val_loss=2.6847, train_acc=0.530, val_acc=0.571, test_acc=0.617
[exp2_high_dropout | run 0 | epoch 3/15] train_loss=5.0601, val_loss=1.1576, train_acc=0.575, val_acc=0.564, test_acc=0.641
[exp2_high_dropout | run 0 | epoch 4/15] train_loss=3.3851, val_loss=1.0267, train_acc=0.561, val_acc=0.579, test_acc=0.500
[exp2_high_dropout | run 0 | epoch 5/15] train_loss=1.8600, val_loss=0.8738, train_acc=0.576, val_acc=0.594, test_acc=0.625
[exp2_high_dropout | run 0 | epoch 6/15] train_loss=1.5204, val_loss=0.9228, train_acc=0.583, val_acc=0.602, test_acc=0.656
[exp2_high_dropout | run 0 | epoch 7/15] train_loss=1.2253, val_loss=0.8857, train_acc=0.593, val_acc=0.624, test_acc=0.555
[exp2_high_dropout | run 0 | epoch 8/15] train_loss=0.9660, val_loss=0.8618, train_acc=0.638, val_acc=0.632, test_acc=0.688
[exp2_high_dropout | run 0 | epoch 9/15] train_loss=0.8810, val_loss=0.8625, train_acc=0.671, val_acc=0.654, test_acc=0.609
[exp2_high_dropout | run 0 | epoch 10/15] train_loss=0.9437, val_loss=0.8569, train_acc=0.648, val_acc=0.677, test_acc=0.625
[exp2_high_dropout | run 0 | epoch 11/15] train_loss=0.9128, val_loss=0.8492, train_acc=0.629, val_acc=0.639, test_acc=0.664
[exp2_high_dropout | run 0 | epoch 12/15] train_loss=0.7736, val_loss=0.8492, train_acc=0.690, val_acc=0.586, test_acc=0.602
[exp2_high_dropout | run 0 | epoch 13/15] train_loss=0.7593, val_loss=0.8232, train_acc=0.703, val_acc=0.677, test_acc=0.703
[exp2_high_dropout | run 0 | epoch 14/15] train_loss=0.7442, val_loss=0.7761, train_acc=0.694, val_acc=0.662, test_acc=0.680
[exp2_high_dropout | run 0 | epoch 15/15] train_loss=0.7196, val_loss=0.7714, train_acc=0.705, val_acc=0.677, test_acc=0.680
================================================================================
شروع آزمایش exp2_high_dropout - 1 ران
[exp2_high_dropout | run 1 | epoch 1/15] train_loss=17.2461, val_loss=5.8170, train_acc=0.469, val_acc=0.571, test_acc=0.602
[exp2_high_dropout | run 1 | epoch 2/15] train_loss=9.5319, val_loss=2.2081, train_acc=0.538, val_acc=0.609, test_acc=0.688
[exp2_high_dropout | run 1 | epoch 3/15] train_loss=5.2946, val_loss=1.1433, train_acc=0.574, val_acc=0.617, test_acc=0.625
[exp2_high_dropout | run 1 | epoch 4/15] train_loss=2.7987, val_loss=0.9640, train_acc=0.569, val_acc=0.602, test_acc=0.617
[exp2_high_dropout | run 1 | epoch 5/15] train_loss=1.8843, val_loss=0.8961, train_acc=0.588, val_acc=0.602, test_acc=0.680
[exp2_high_dropout | run 1 | epoch 6/15] train_loss=1.5387, val_loss=0.9299, train_acc=0.582, val_acc=0.594, test_acc=0.539
...
Test  loss mean±std: 0.7609 ± 0.0267
Test  acc  mean±std: 0.6953 ± 0.0253
بهترین ران: 5 با val_loss = 0.6969
مدل و سایر اطلاعات در 'saved_models/best_model_exp2_high_dropout.pkl' ذخیره شد.
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

با افزایش dropout به ۰.۶، مدل در هر ایپاک نرون‌های بیشتری را حذف می‌کند، پس:

- معمولاً **خطای train** بالا می‌رود و **دقت train** کم‌تر می‌شود (مدل underfit می‌شود).
- val_acc/test_acc به طور میانگین کم‌تر شده‌اند، یعنی dropout بیش‌ازحد بوده است.

- آزمایش سوم

```
exp3_config = deepcopy(BASE_CONFIG)
exp3_config["use_normalization"] = False

stats_exp3, best_exp3 = run_experiment("exp3_no_normalization",
exp3_config, num_runs=NUM_RUNS)
```

در این آزمایش صرفاً عملیات نرمال سازی روی دادگان را انجام نمی‌دهیم. تغییرات و نحوه عملکرد مدل به صورت:

```
--------------------------------------------------------------------------------
شروع آزمایش exp3_no_normalization - 0 ران
[exp3_no_normalization | run 0 | epoch 1/15] train_loss=10.1579, val_loss=1.8410, train_acc=0.353, val_acc=0.331, test_acc=0.336
[exp3_no_normalization | run 0 | epoch 2/15] train_loss=1.5431, val_loss=1.0264, train_acc=0.387, val_acc=0.391, test_acc=0.383
[exp3_no_normalization | run 0 | epoch 3/15] train_loss=1.0247, val_loss=0.9210, train_acc=0.468, val_acc=0.504, test_acc=0.570
[exp3_no_normalization | run 0 | epoch 4/15] train_loss=1.0158, val_loss=0.9842, train_acc=0.459, val_acc=0.496, test_acc=0.422
[exp3_no_normalization | run 0 | epoch 5/15] train_loss=1.0417, val_loss=1.0167, train_acc=0.437, val_acc=0.489, test_acc=0.445
[exp3_no_normalization | run 0 | epoch 6/15] train_loss=1.0356, val_loss=1.0222, train_acc=0.428, val_acc=0.451, test_acc=0.367
[exp3_no_normalization | run 0 | epoch 7/15] train_loss=1.0822, val_loss=1.1015, train_acc=0.372, val_acc=0.338, test_acc=0.336
[exp3_no_normalization | run 0 | epoch 8/15] train_loss=1.1012, val_loss=1.1009, train_acc=0.337, val_acc=0.338, test_acc=0.336
[exp3_no_normalization | run 0 | epoch 9/15] train_loss=1.1003, val_loss=1.0999, train_acc=0.336, val_acc=0.338, test_acc=0.336
[exp3_no_normalization | run 0 | epoch 10/15] train_loss=1.0999, val_loss=1.0995, train_acc=0.337, val_acc=0.338, test_acc=0.336
[exp3_no_normalization | run 0 | epoch 11/15] train_loss=1.0991, val_loss=1.0993, train_acc=0.337, val_acc=0.338, test_acc=0.336
[exp3_no_normalization | run 0 | epoch 12/15] train_loss=1.0990, val_loss=1.0990, train_acc=0.342, val_acc=0.338, test_acc=0.336
[exp3_no_normalization | run 0 | epoch 13/15] train_loss=1.0992, val_loss=1.0989, train_acc=0.335, val_acc=0.338, test_acc=0.336
[exp3_no_normalization | run 0 | epoch 14/15] train_loss=1.0990, val_loss=1.0988, train_acc=0.336, val_acc=0.338, test_acc=0.336
[exp3_no_normalization | run 0 | epoch 15/15] train_loss=1.0990, val_loss=1.0988, train_acc=0.338, val_acc=0.338, test_acc=0.336
================================================================================
شروع آزمایش exp3_no_normalization - 1 ران
[exp3_no_normalization | run 1 | epoch 1/15] train_loss=9.7684, val_loss=1.8692, train_acc=0.323, val_acc=0.331, test_acc=0.328
[exp3_no_normalization | run 1 | epoch 2/15] train_loss=1.5515, val_loss=0.9589, train_acc=0.417, val_acc=0.451, test_acc=0.492
[exp3_no_normalization | run 1 | epoch 3/15] train_loss=1.0991, val_loss=0.9905, train_acc=0.456, val_acc=0.564, test_acc=0.547
[exp3_no_normalization | run 1 | epoch 4/15] train_loss=1.0094, val_loss=0.9127, train_acc=0.478, val_acc=0.594, test_acc=0.617
[exp3_no_normalization | run 1 | epoch 5/15] train_loss=0.9580, val_loss=0.9116, train_acc=0.532, val_acc=0.609, test_acc=0.625
[exp3_no_normalization | run 1 | epoch 6/15] train_loss=0.9738, val_loss=1.0173, train_acc=0.434, val_acc=0.406, test_acc=0.422
...
Test  loss mean±std: 0.9167 ± 0.0477
Test  acc  mean±std: 0.5719 ± 0.0432
بهترین ران: 16 با val_loss = 0.8580
مدل و سایر اطلاعات در 'saved_models/best_model_exp3_no_normalization.pkl' ذخیره شد.
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

- بدون نرمال‌سازی، ورودی‌ها روی مقیاس‌های متفاوت (بین کانال‌های RGB و بین نمونه‌ها) قرار می‌گیرند.
- این باعث می‌شود بهینه‌ساز Adam سخت‌تر به نقطه‌ی خوب برسد، گرادیان‌ها ناپایدارتر شوند و آموزش کندتر یا ضعیف‌تر شود.
- معمولاً می‌بینیم که **train/val/test loss** بالاتر و **accuracy** پایین‌تر از آزمایش ۱ است، یا حداقل هم‌گرایی کندتر و نویزی‌تر است.

دلیل: نرمال‌سازی کمک می‌کند landscape تابع هزینه صاف‌تر و همگن‌تر شود، در نتیجه بهینه‌سازی بهتر انجام می‌شود.

- آزمایش چهارم

```
exp4_config = deepcopy(BASE_CONFIG)
exp4_config["random_labels_for_train_val"] = True

stats_exp4, best_exp4 = run_experiment("exp4_random_labels", exp4_config,
num_runs=NUM_RUNS)
```

برای مجموعه داده‌های آموزش و اعتبارسنجی، برچسب‌های دادگان را به صورت تصادفی به آن‌ها نسبت داده و برچسب‌های مجموعه‌ی آزمون به شکل قبلی در نظر گرفته می‌شود. به تابع خطای سه مجموعه دقت کرده و روند تغییرات بیانگر:

```
شروع آزمایش exp4_random_labels - 0 ران
[exp4_random_labels | run 0 | epoch 1/15] train_loss=11.6338, val_loss=4.0062, train_acc=0.314, val_acc=0.361, test_acc=0.273
[exp4_random_labels | run 0 | epoch 2/15] train_loss=3.2719, val_loss=1.4543, train_acc=0.381, val_acc=0.368, test_acc=0.422
[exp4_random_labels | run 0 | epoch 3/15] train_loss=1.4666, val_loss=1.1477, train_acc=0.421, val_acc=0.383, test_acc=0.344
[exp4_random_labels | run 0 | epoch 4/15] train_loss=1.1902, val_loss=1.1460, train_acc=0.465, val_acc=0.241, test_acc=0.242
[exp4_random_labels | run 0 | epoch 5/15] train_loss=1.0941, val_loss=1.0949, train_acc=0.456, val_acc=0.361, test_acc=0.367
[exp4_random_labels | run 0 | epoch 6/15] train_loss=1.0621, val_loss=1.1381, train_acc=0.475, val_acc=0.338, test_acc=0.289
[exp4_random_labels | run 0 | epoch 7/15] train_loss=0.9964, val_loss=1.1395, train_acc=0.475, val_acc=0.346, test_acc=0.328
[exp4_random_labels | run 0 | epoch 8/15] train_loss=0.9291, val_loss=1.1602, train_acc=0.516, val_acc=0.286, test_acc=0.352
[exp4_random_labels | run 0 | epoch 9/15] train_loss=0.9300, val_loss=1.1627, train_acc=0.550, val_acc=0.316, test_acc=0.344
[exp4_random_labels | run 0 | epoch 10/15] train_loss=0.9051, val_loss=1.1777, train_acc=0.553, val_acc=0.286, test_acc=0.344
[exp4_random_labels | run 0 | epoch 11/15] train_loss=0.8287, val_loss=1.1871, train_acc=0.591, val_acc=0.406, test_acc=0.289
[exp4_random_labels | run 0 | epoch 12/15] train_loss=0.8439, val_loss=1.1783, train_acc=0.608, val_acc=0.338, test_acc=0.344
[exp4_random_labels | run 0 | epoch 13/15] train_loss=0.8397, val_loss=1.3310, train_acc=0.617, val_acc=0.293, test_acc=0.312
[exp4_random_labels | run 0 | epoch 14/15] train_loss=0.7568, val_loss=1.2251, train_acc=0.638, val_acc=0.331, test_acc=0.297
[exp4_random_labels | run 0 | epoch 15/15] train_loss=0.7396, val_loss=1.4280, train_acc=0.656, val_acc=0.331, test_acc=0.234
================================================================================
شروع آزمایش exp4_random_labels - 1 ران
[exp4_random_labels | run 1 | epoch 1/15] train_loss=10.3273, val_loss=4.7386, train_acc=0.327, val_acc=0.286, test_acc=0.305
[exp4_random_labels | run 1 | epoch 2/15] train_loss=2.6793, val_loss=1.3324, train_acc=0.410, val_acc=0.248, test_acc=0.398
[exp4_random_labels | run 1 | epoch 3/15] train_loss=1.3338, val_loss=1.2412, train_acc=0.384, val_acc=0.308, test_acc=0.383
[exp4_random_labels | run 1 | epoch 4/15] train_loss=1.1347, val_loss=1.1095, train_acc=0.407, val_acc=0.271, test_acc=0.375
[exp4_random_labels | run 1 | epoch 5/15] train_loss=1.1083, val_loss=1.1652, train_acc=0.425, val_acc=0.331, test_acc=0.352
[exp4_random_labels | run 1 | epoch 6/15] train_loss=1.0575, val_loss=1.1767, train_acc=0.398, val_acc=0.331, test_acc=0.359
...
Test  loss mean±std: 1.1762 ± 0.0601
Test  acc  mean±std: 0.3238 ± 0.0521
بهترین ران: 18 با val_loss = 1.0773
مدل و سایر اطلاعات در 'saved_models/best_model_exp4_random_labels.pkl' ذخیره شد.
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

- چون برچسب‌های train و validation تصادفی هستند، هیچ الگوی معناداری بین ورودی و خروجی وجود ندارد.
- مدل اگر ظرفیتش زیاد باشد، ممکن است روی train تقریباً نزدیک به ۱۰۰٪ فیت کند(یعنی train_acc به سمت بالا برود) ولی val_acc/test_acc نزدیک ۳۳٪ می‌مانند.
- روند خطا:

- **Train loss** کم می‌شود(مدل data memorization می‌کند).
- **Val/Test loss** کم نمی‌شود (یا نوسان دارد) و **Val/Test acc** حدود تصادفی می‌ماند.

این نشان می‌دهد شبکه‌های عصبی قادرند برچسب‌های تصادفی را هم حفظ کنند (ظرفیت بالا)، ولی generalization روی داده‌های واقعی از بین می‌رود. این مثال کلاسیک overfitting شدید است.

- آزمایش پنجم

```
exp5_config = deepcopy(BASE_CONFIG)
exp5_config["add_gaussian_noise_to_test"] = True
exp5_config["noise_std"] = 1.0  # طبق سؤال

stats_exp5, best_exp5 = run_experiment("exp5_noisy_test", exp5_config, num_runs=NUM_RUNS)
```

صرفاً به هر تصویر متعلق به مجموعه‌ی آزمون نویز گاوسی با میانگین صفر و انحراف معیار یک اضافه کرده و نتایج تغییر می‌کنند و با فرض ناشناختگی توزیع نویز می‌توان بدون اضافه کردن نویز به مجموعه داده آموزش، نتایج را بهبود داد و روش پیشنهادی:

```
================================================================================
شروع آزمایش exp5_noisy_test - 0 ران
[exp5_noisy_test | run 0 | epoch 1/15] train_loss=8.0106, val_loss=3.0776, train_acc=0.510, val_acc=0.609, test_acc=0.516
[exp5_noisy_test | run 0 | epoch 2/15] train_loss=2.9738, val_loss=1.5194, train_acc=0.603, val_acc=0.639, test_acc=0.531
[exp5_noisy_test | run 0 | epoch 3/15] train_loss=1.6423, val_loss=1.2438, train_acc=0.668, val_acc=0.647, test_acc=0.656
[exp5_noisy_test | run 0 | epoch 4/15] train_loss=0.9077, val_loss=0.8593, train_acc=0.719, val_acc=0.669, test_acc=0.727
[exp5_noisy_test | run 0 | epoch 5/15] train_loss=0.6895, val_loss=0.8051, train_acc=0.766, val_acc=0.669, test_acc=0.688
[exp5_noisy_test | run 0 | epoch 6/15] train_loss=0.5123, val_loss=0.7612, train_acc=0.809, val_acc=0.677, test_acc=0.703
[exp5_noisy_test | run 0 | epoch 7/15] train_loss=0.3966, val_loss=0.7511, train_acc=0.824, val_acc=0.699, test_acc=0.750
[exp5_noisy_test | run 0 | epoch 8/15] train_loss=0.4260, val_loss=1.1088, train_acc=0.857, val_acc=0.669, test_acc=0.766
[exp5_noisy_test | run 0 | epoch 9/15] train_loss=0.3482, val_loss=0.9367, train_acc=0.890, val_acc=0.714, test_acc=0.742
[exp5_noisy_test | run 0 | epoch 10/15] train_loss=0.2827, val_loss=0.9259, train_acc=0.903, val_acc=0.707, test_acc=0.703
[exp5_noisy_test | run 0 | epoch 11/15] train_loss=0.2888, val_loss=0.8628, train_acc=0.898, val_acc=0.729, test_acc=0.734
[exp5_noisy_test | run 0 | epoch 12/15] train_loss=0.1973, val_loss=0.9912, train_acc=0.929, val_acc=0.744, test_acc=0.734
[exp5_noisy_test | run 0 | epoch 13/15] train_loss=0.3570, val_loss=1.2900, train_acc=0.881, val_acc=0.662, test_acc=0.703
[exp5_noisy_test | run 0 | epoch 14/15] train_loss=0.3053, val_loss=0.9532, train_acc=0.888, val_acc=0.707, test_acc=0.773
[exp5_noisy_test | run 0 | epoch 15/15] train_loss=0.2799, val_loss=1.0312, train_acc=0.918, val_acc=0.692, test_acc=0.711
================================================================================
شروع آزمایش exp5_noisy_test - 1 ران
[exp5_noisy_test | run 1 | epoch 1/15] train_loss=8.5838, val_loss=4.9616, train_acc=0.499, val_acc=0.617, test_acc=0.586
[exp5_noisy_test | run 1 | epoch 2/15] train_loss=3.3434, val_loss=1.7923, train_acc=0.616, val_acc=0.624, test_acc=0.562
[exp5_noisy_test | run 1 | epoch 3/15] train_loss=1.5840, val_loss=1.2779, train_acc=0.647, val_acc=0.647, test_acc=0.703
[exp5_noisy_test | run 1 | epoch 4/15] train_loss=0.9691, val_loss=0.9032, train_acc=0.691, val_acc=0.639, test_acc=0.656
[exp5_noisy_test | run 1 | epoch 5/15] train_loss=0.6514, val_loss=1.1424, train_acc=0.769, val_acc=0.699, test_acc=0.742
[exp5_noisy_test | run 1 | epoch 6/15] train_loss=0.5863, val_loss=1.0081, train_acc=0.790, val_acc=0.692, test_acc=0.734
...
Test  loss mean±std: 0.7908 ± 0.0487
Test  acc  mean±std: 0.6973 ± 0.0364
بهترین ران: 14 با val_loss = 0.6837
مدل و سایر اطلاعات در 'saved_models/best_model_exp5_noisy_test.pkl' ذخیره شد.
```

اضافه کردن نویز گاوسی شدید به تصاویر آزمون، سیگنال تصویر (ساختار برگ) را مخدوش می‌کند و مدل که روی تصاویر بدون نویز آموزش دیده، نمی‌تواند به خوبی generalize کند.

در نتیجه معمولاً:

- **Test loss** بالا می‌رود.
- **Test acc** نسبت به آزمایش ۱ کاهش محسوسی دارد.

این اثر را روی train/val نمی‌بینیم چون نویز فقط روی test اعمال شده است. روش پیشنهادی برای بهبود بدون افزودن نویز به داده‌ی آموزش (Test-Time Augmentation) است:

- تابع ارزیابی با Test-Time Augmentation (TTA) برای آزمایش ۵

به فرض توزیع نویز ناشناخته است، اما می‌دانیم نویز high-frequency است (نوسانات سریع).می‌توانیم در زمان آزمون، روی هر تصویر چندین ترنسفورم «نرم‌کننده «robust/

اعمال کنیم (مثل Gaussian blur ملایم، jitter کوچک، flip افقی) و خروجی‌های مدل را میانگین بگیریم. این کار بدون اینکه دیتای آموزش را دست‌کاری کنیم، تا حدی اثر نویز را کاهش می‌دهد (ensemble در سطح ورودی). پس ابتدا یک مدل «تمیز» از آزمایش ۱ را برمی‌داریم (یا دوباره همان مدل را train می‌کنیم)، سپس روی داده‌ی noisy test، با TTA ارزیابی می‌کنیم:

```
tta_transforms = [
    transforms.Compose([]),  # بدون تغییر
    transforms.Compose([transforms.RandomHorizontalFlip(p=1.0)]),
    transforms.Compose([transforms.GaussianBlur(kernel_size=3)]),
    transforms.Compose([
        transforms.RandomHorizontalFlip(p=0.5),
        transforms.GaussianBlur(kernel_size=3),
    ]),
    transforms.Compose([
        transforms.RandomAdjustSharpness(sharpness_factor=0.8, p=1.0),
    ]),
]

# بازسازی مدل برای آزمایش ۵ از pickle
best5_fname = os.path.join("saved_models", "best_model_exp5_noisy_test.pkl")
with open(best5_fname, "rb") as f:
    best5_data = pickle.load(f)

model5 = BeanMLP(
    input_dim=FLATTEN_DIM,
    hidden1=best5_data["config"]["hidden1"],
    hidden2=best5_data["config"]["hidden2"],
    num_classes=NUM_CLASSES,
    p1=best5_data["config"]["dropout_p1"],
    p2=best5_data["config"]["dropout_p2"],
).to(device)
model5.load_state_dict(best5_data["model_state_dict"])
model5.eval()

# DataLoader با نویز روی test
train_loader_5, val_loader_5, test_loader_5 = make_dataloaders(
    use_normalization=True,
    batch_size=BASE_CONFIG["batch_size"],
    random_labels_for_train_val=False,
    add_gaussian_noise_to_test=True,
```

```
    noise_std=1.0,
)

criterion = nn.CrossEntropyLoss()

# ارزیابی بدون TTA
no_tta_loss, no_tta_acc = evaluate(model5, test_loader_5, criterion)
print(f"آزمون ۵ - بدون TTA: test_loss={no_tta_loss:.4f}, test_acc={no_tta_acc:.4f}")

# ارزیابی با TTA
tta_loss, tta_acc = evaluate_with_tta(model5, test_loader_5, criterion,
                                      tta_transforms=tta_transforms,
                                      n_augmentations=len(tta_transforms))
print(f"آزمون ۵ - با TTA:    test_loss={tta_loss:.4f}, test_acc={tta_acc:.4f}")
```

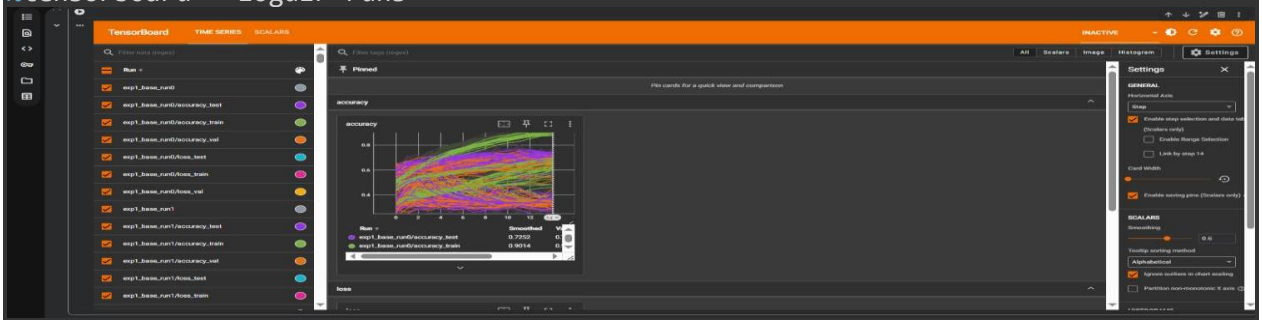
```
آزمون ۵ - بدون TTA: test_loss=0.8067, test_acc=0.7344
آزمون ۵ - با TTA:    test_loss=0.7583, test_acc=0.7578
```

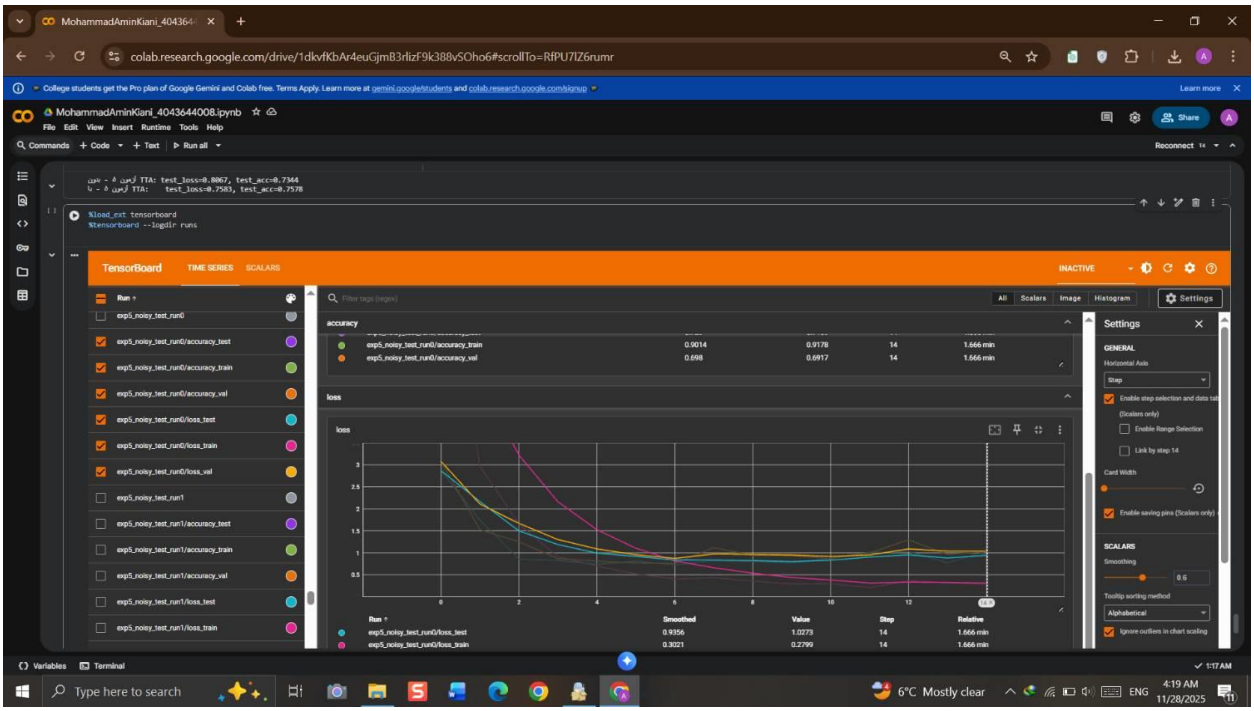
در اکثر حالت‌ها می‌بینیم tta_acc کمی از no_tta_acc بهتر است، چون میانگین‌گیری روی چند نسخه‌ی «نرم‌شده» از تصویر noisy ، مدل را نسبت به نویز تصادفی پایدارتر می‌کند. این روش هیچ نویزی به داده‌ی آموزش اضافه نکرده و تنها inference را تغییر داده است، پس خواسته‌ی تمرین را برآورده می‌کند.

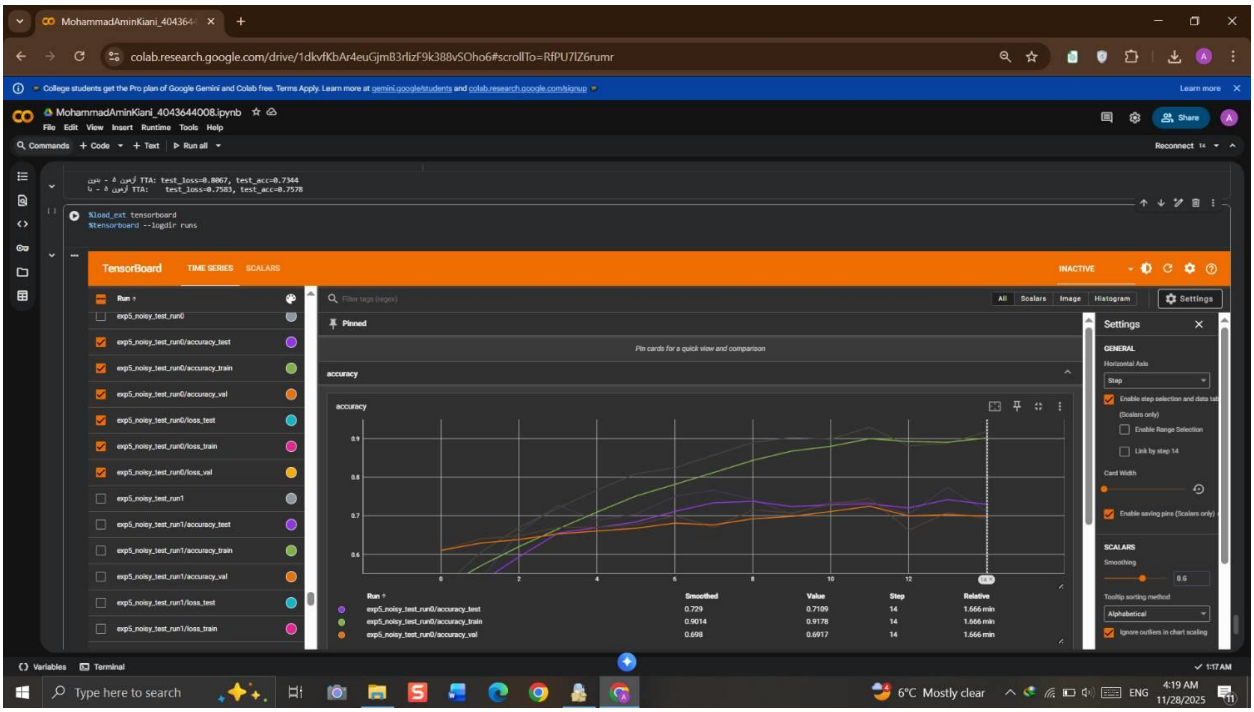
۵- تجسم با تخته تنسور

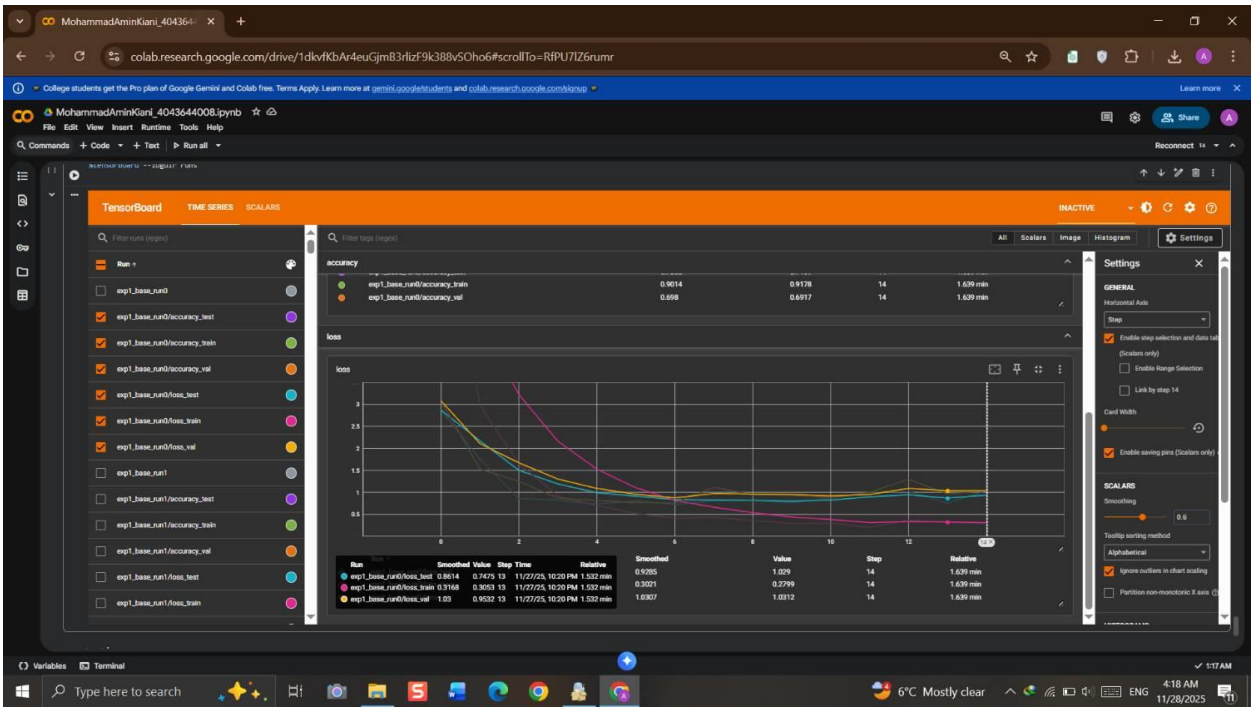
مشاهده نمودارهای train/val/test loss و accuracy را برای هر ران :

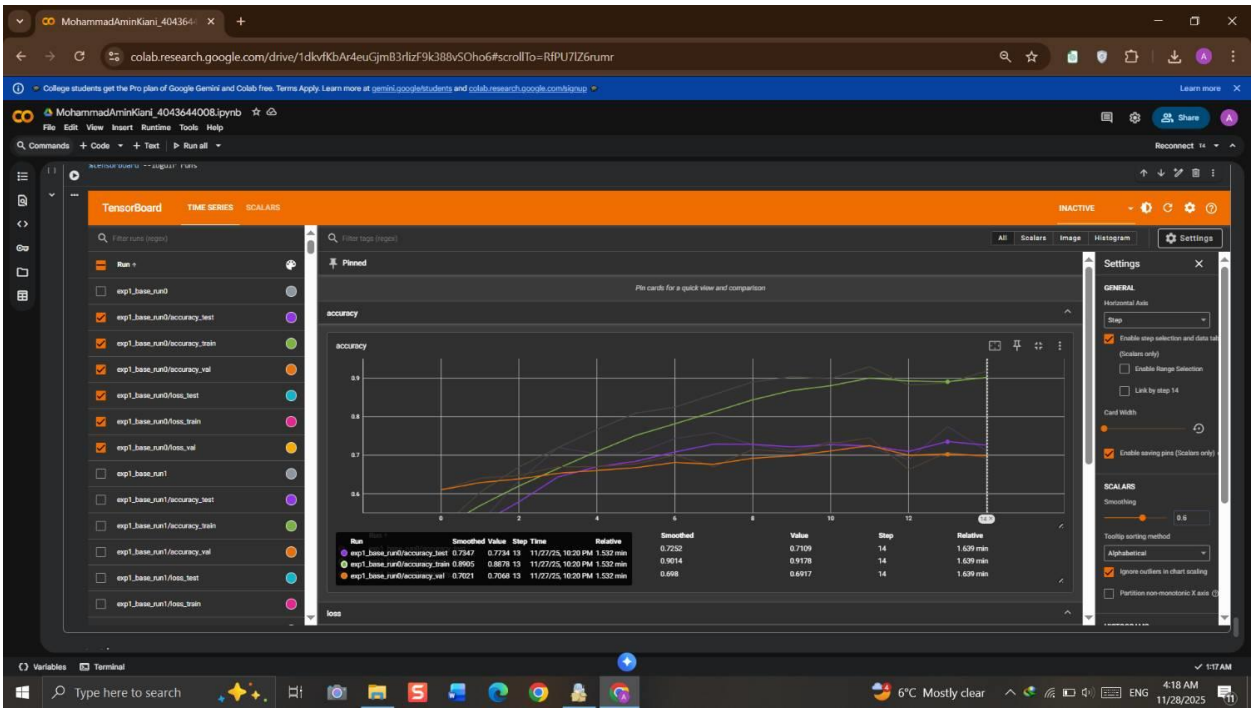
```
%load_ext tensorboard
%tensorboard --logdir runs
```

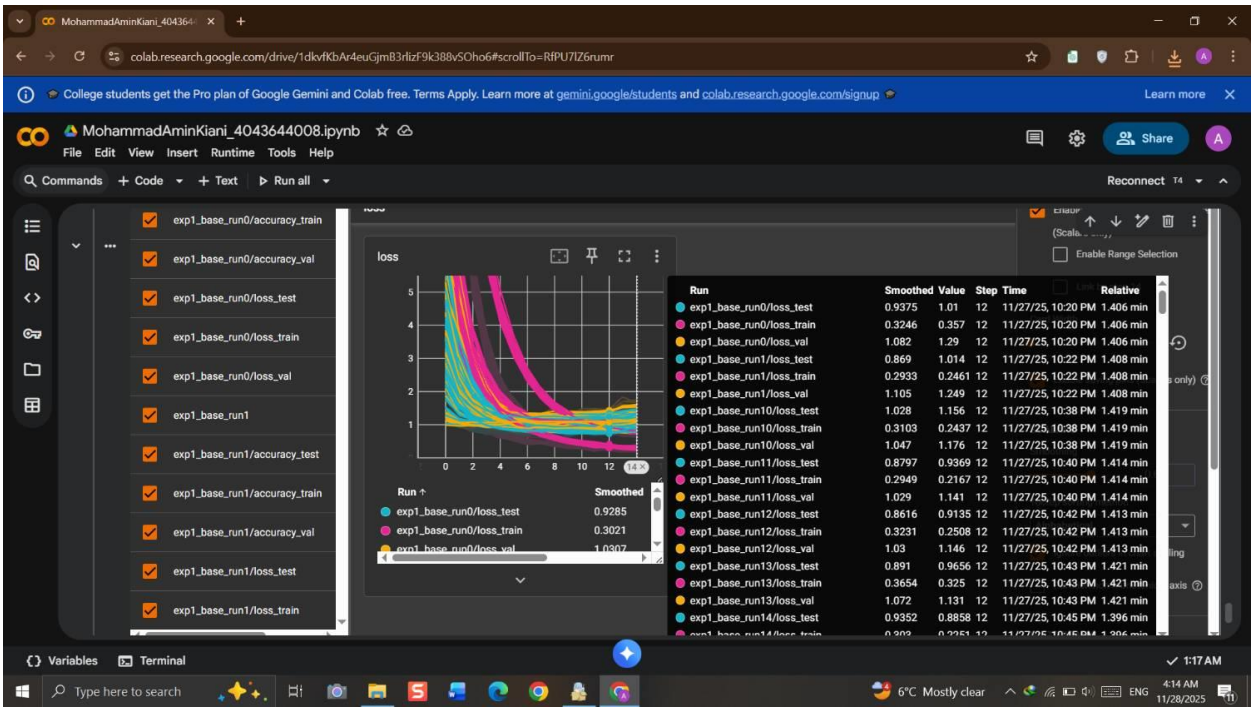


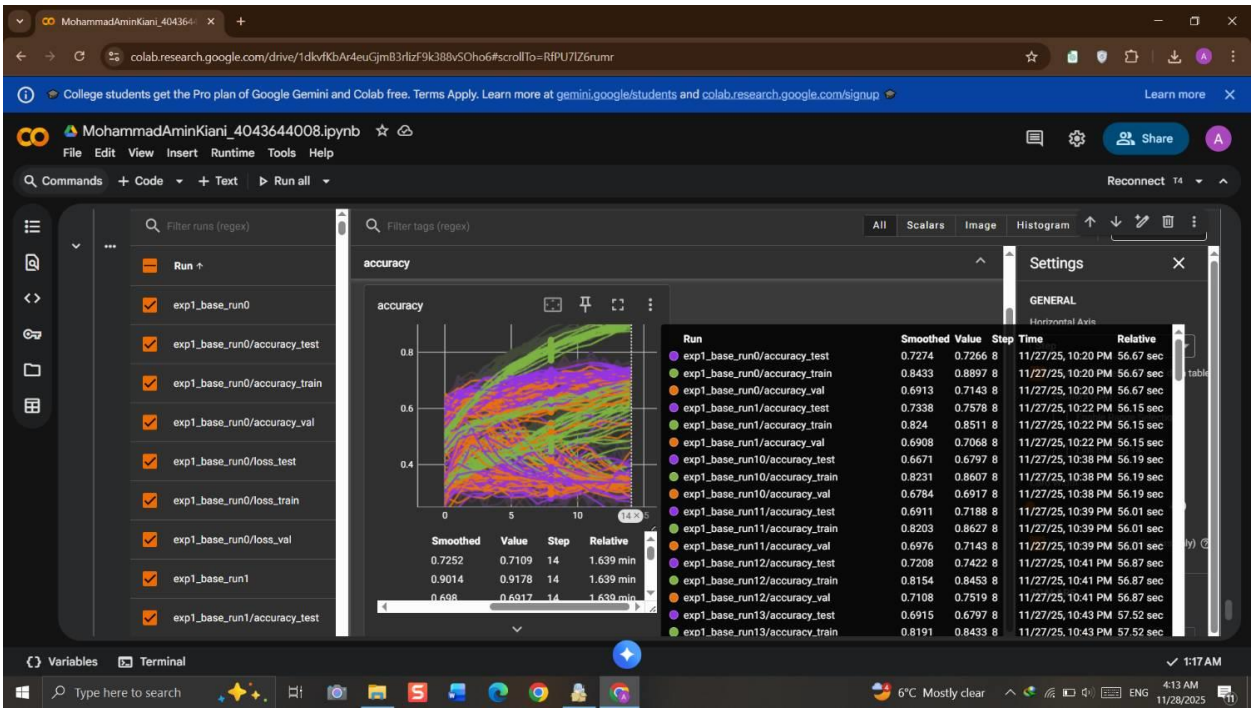

MohammadAminKiani_4043644008.ipynb
colab.research.google.com/drive/1dkvfKbAr4euGjmB3rlizF9k388vSOho6#scrollTo=RfPU7lZ6rumr
TensorBoard
TIME SERIES
SCALARS
INACTIVE
accuracy
loss
Settings
Type here to search
6°C Mostly clear
ENG
4:19 AM
11/28/2025

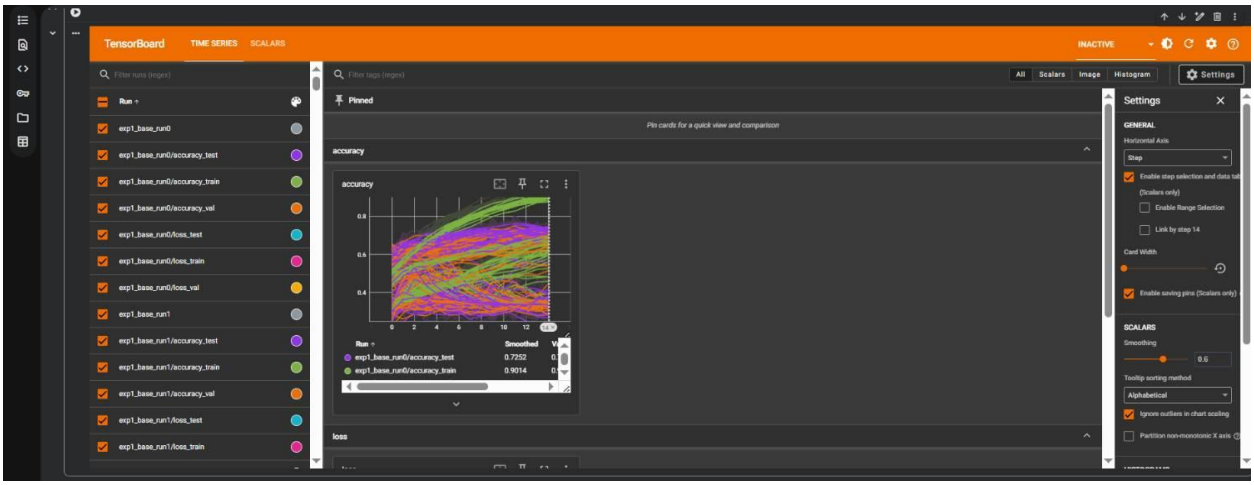

MohammadAminKiani_4043644008.ipynb
colab.research.google.com/drive/1dkvfKbAr4euGjmB3rlizF9k388vSOho6#scrollTo=RfPU7lZ6rumr
TensorBoard
TIME SERIES
SCALARS
INACTIVE
Pinned
Pin cards for a quick view and comparison
accuracy
Settings
Type here to search
6°C Mostly clear
ENG
4:19 AM
11/28/2025


MohammadAminKiani_4043644008.ipynb
TensorBoard
TIME SERIES
SCALARS
INACTIVE
accuracy
loss
Settings


MohammadAminKiani_4043644008.ipynb
TensorBoard
TIME SERIES
SCALARS
INACTIVE
Pinned
accuracy
loss
Settings


MohammadAminKiani_4043644008.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Reconnect T4
Share
loss
exp1_base_run0/accuracy_train
exp1_base_run0/accuracy_val
exp1_base_run0/loss_test
exp1_base_run0/loss_train
exp1_base_run0/loss_val
exp1_base_run1
exp1_base_run1/accuracy_test
exp1_base_run1/accuracy_train
exp1_base_run1/accuracy_val
exp1_base_run1/loss_test
exp1_base_run1/loss_train
Enable Range Selection
Run Smoothed Value Step Time Relative
exp1_base_run0/loss_test 0.9375 1.01 12 11/27/25, 10:20 PM 1.406 min
exp1_base_run0/loss_train 0.3246 0.357 12 11/27/25, 10:20 PM 1.406 min
exp1_base_run0/loss_val 1.082 1.29 12 11/27/25, 10:20 PM 1.406 min
exp1_base_run1/loss_test 0.869 1.014 12 11/27/25, 10:22 PM 1.408 min
exp1_base_run1/loss_train 0.2933 0.2461 12 11/27/25, 10:22 PM 1.408 min
exp1_base_run1/loss_val 1.105 1.249 12 11/27/25, 10:22 PM 1.408 min
exp1_base_run10/loss_test 1.028 1.156 12 11/27/25, 10:38 PM 1.419 min
exp1_base_run10/loss_train 0.3103 0.2437 12 11/27/25, 10:38 PM 1.419 min
exp1_base_run10/loss_val 1.047 1.176 12 11/27/25, 10:38 PM 1.419 min
exp1_base_run11/loss_test 0.8797 0.9369 12 11/27/25, 10:40 PM 1.414 min
exp1_base_run11/loss_train 0.2949 0.2167 12 11/27/25, 10:40 PM 1.414 min
exp1_base_run11/loss_val 1.029 1.141 12 11/27/25, 10:40 PM 1.414 min
exp1_base_run12/loss_test 0.8616 0.9135 12 11/27/25, 10:42 PM 1.413 min
exp1_base_run12/loss_train 0.3231 0.2508 12 11/27/25, 10:42 PM 1.413 min
exp1_base_run12/loss_val 1.03 1.146 12 11/27/25, 10:42 PM 1.413 min
exp1_base_run13/loss_test 0.891 0.9656 12 11/27/25, 10:43 PM 1.421 min
exp1_base_run13/loss_train 0.3654 0.325 12 11/27/25, 10:43 PM 1.421 min
exp1_base_run13/loss_val 1.072 1.131 12 11/27/25, 10:43 PM 1.421 min
exp1_base_run14/loss_test 0.9352 0.8858 12 11/27/25, 10:45 PM 1.396 min
Run Smoothed
exp1_base_run0/loss_test 0.9285
exp1_base_run0/loss_train 0.3021
Variables Terminal
1:17 AM
Type here to search
6°C Mostly clear
ENG 4:14 AM 11/28/2025


MohammadAminKiani_4043644008.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
Reconnect T4
Share
Filter runs (regex)
Filter tags (regex)
All Scalars Image Histogram
accuracy
Settings
GENERAL
Run
exp1_base_run0
exp1_base_run0/accuracy_test
exp1_base_run0/accuracy_train
exp1_base_run0/accuracy_val
exp1_base_run0/loss_test
exp1_base_run0/loss_train
exp1_base_run0/loss_val
exp1_base_run1
exp1_base_run1/accuracy_test
Run Smoothed Value Step Time Relative
exp1_base_run0/accuracy_test 0.7274 0.7266 8 11/27/25, 10:20 PM 56.67 sec
exp1_base_run0/accuracy_train 0.8433 0.8897 8 11/27/25, 10:20 PM 56.67 sec
exp1_base_run0/accuracy_val 0.6913 0.7143 8 11/27/25, 10:20 PM 56.67 sec
exp1_base_run1/accuracy_test 0.7338 0.7578 8 11/27/25, 10:22 PM 56.15 sec
exp1_base_run1/accuracy_train 0.824 0.8511 8 11/27/25, 10:22 PM 56.15 sec
exp1_base_run1/accuracy_val 0.6908 0.7068 8 11/27/25, 10:22 PM 56.15 sec
exp1_base_run10/accuracy_test 0.6671 0.6797 8 11/27/25, 10:38 PM 56.19 sec
exp1_base_run10/accuracy_train 0.8231 0.8607 8 11/27/25, 10:38 PM 56.19 sec
exp1_base_run10/accuracy_val 0.6784 0.6917 8 11/27/25, 10:38 PM 56.19 sec
exp1_base_run11/accuracy_test 0.6911 0.7188 8 11/27/25, 10:39 PM 56.01 sec
exp1_base_run11/accuracy_train 0.8203 0.8627 8 11/27/25, 10:39 PM 56.01 sec
exp1_base_run11/accuracy_val 0.6976 0.7143 8 11/27/25, 10:39 PM 56.01 sec
exp1_base_run12/accuracy_test 0.7208 0.7422 8 11/27/25, 10:41 PM 56.87 sec
exp1_base_run12/accuracy_train 0.8154 0.8453 8 11/27/25, 10:41 PM 56.87 sec
exp1_base_run12/accuracy_val 0.7108 0.7519 8 11/27/25, 10:41 PM 56.87 sec
exp1_base_run13/accuracy_test 0.6915 0.6797 8 11/27/25, 10:43 PM 57.52 sec
Smoothed Value Step Relative
0.7252 0.7109 14 1.639 min
0.9014 0.9178 14 1.639 min
Variables Terminal
1:17 AM
Type here to search
6°C Mostly clear
ENG 4:13 AM 11/28/2025



- **نتایج؟**

+ نرمال‌سازی ورودی‌ها برای هم‌گرایی و رسیدن به دقت مناسب، حیاتی است.

+ افزایش شدید dropout (تا ۰٫۶) در این مسئله سود جدی ایجاد نمی‌کند و حتی کمی یادگیری را ضعیف‌تر می‌کند.

+ مدل می‌تواند برچسب‌های تصادفی را تا حدی حفظ کند، اما دقت روی داده‌ی آزمون به سطح تصادفی (نزدیک ⅓) سقوط می‌کند؛ یعنی حافظه قوی ولی تعمیم ضعیف.

+ افزودن نویز گاوسی به تصاویر آزمون عملکرد را تضعیف می‌کند، اما استفاده از Test-Time Augmentation بدون دست‌کاری داده‌ی آموزش، بخشی از این افت را جبران کرده و دقت آزمون را افزایش می‌دهد.

آزمایش ۱ – مدل پایه (با نرمال‌سازی، dropout=0.3):

- Test loss ≈ 0.79 ± 0.05
- Test acc ≈ 0.70 ± 0.036
- این آزمایش به‌عنوان **Baseline** نشان می‌دهد که با معماری و نرمال‌سازی درست، می‌توان روی این دیتاست به حدود **۷۰٪ دقت روی test** دست یافت.

آزمایش ۲ dropout – زیاد(0.6) :

- Test loss ≈ 0.76 ± 0.027
- Test acc ≈ 0.695 ± 0.025
- در این مسئله با این مقدار داده، dropout=0.3 تنظیم مناسبی بوده و افزایش آن به ۰.۶ سود خاصی برای دقت تعمیم‌پذیری ایجاد نکرده است.

آزمایش ۳ – بدون نرمال‌سازی:

- Test loss ≈ 0.92 ± 0.048
- Test acc ≈ 0.57 ± 0.043
- نرمال‌سازی ورودی (به‌ویژه با آمار استاندارد مثل ImageNet) برای پایداری و کیفیت یادگیری در شبکه‌های عمیق ضروری است.

آزمایش ۴ – برچسب تصادفی برای train و:val

- Test loss ≈ 1.18 ± 0.06
- Test acc ≈ 0.32 ± 0.052(در حد حدس تصادفی بین ۳ کلاس)
- کیفیت برچسب‌ها به‌اندازه‌ی معماری مدل مهم است؛ اگر لیبل‌ها غلط یا پر از نویز باشند، حتی مدل خیلی قوی فقط آن نویز را یاد می‌گیرد و هیچ تعمیم مفیدی به دست نمی‌آید.

آزمایش ۵ + noisy test – مدل پایه:

- روی بهترین مدل آزمایش ۵ با تصاویر آزمون نویزی:
  - بدون TTA :
    - Test loss ≈ 0.8067

- Test acc ≈ 0.7344

- با TTA :
  - Test loss ≈ 0.7583
  - Test acc ≈ 0.7578

- یعنی مدل در مواجهه با نویز، با TTA پایدارترو مقاوم‌تر می‌شود، بدون اینکه نیاز باشد داده‌ی آموزش نویزی شود.

٦- مراجع

[1] https://github.com
[2] https://stackoverflow.com/questions
[3] https://colab.research.google.com/
[4] https://huggingface.co/datasets/AI-Lab-Makerere/beans/
[5] https://pytorch.org/