数据集：五类任务，六个数据集

任务名称	数据集	特点
GEAC (通用加密应用分类)	Cross-Platform (iOS/Android)	200+ 应用，类别不均衡
EMC (加密恶意流量分类)	USTC-TFC	包含10类良性与10类恶意流量
ETCV (VPN流量分类)	ISCX-VPN-Service/App	识别VPN通信
EACT (Tor流量分类)	ISCX-Tor	匿名通信，难度高
EAC-1.3 (TLS 1.3流量分类)	CSTNET-TLS 1.3 (自建)	120个真实应用流量

Cross-Platform (iOS/Android)：https://github.com/mohamedali92/cross-platform-apps

ISCX-VPN-Service/APP，USTC-TFC：https://drive.google.com/file/d/1F09zxln9iFg2HWoqc6m4LKFhYK7cDQv_/view

ISCX-Tor：https://www.unb.ca/cic/datasets/tor.html

CSTNET-TLS 1.3：https://github.com/linwhitehat/ET-BERT/tree/main/datasets/CSTNET-TLS%201.3

4.1.2 数据预处理 (Data Pre-processing)

为了避免非内容特征（如端口号、IP头部）引入偏差，作者进行了严格的清洗与截断处理：

1. **剔除无关包**：删除 ARP（地址解析协议）和 DHCP（动态主机配置协议）包；
2. **去除头部偏差**：去掉以太网头、IP头和 TCP 端口信息，防止模型利用“显著标识符”而非内容；
3. **采样平衡**：
    - 每类最多保留 500 条流（flows）和 5,000 个包（packets）；
    - 以 8:1:1 的比例划分为训练、验证与测试集；
4. **输入格式统一**：每个输入由最多 5 个连续包拼接形成一个 datagram token 序列（M=5）。

4.1.3 评价指标与实现细节 (Evaluation Metrics and Implementation Details)

评价指标

采用四个常见分类指标：

- **Accuracy (AC)** 准确率
- **Precision (PR)** 精确率
- **Recall (RC)** 召回率
- **F1-score (F1)** 综合指标

> *为应对类别不平衡问题，使用* ***Macro Average****（对每类独立计算，再平均）。*

实现细节

- 预训练阶段：
    - 数据规模：30GB 无标签流量（15GB 来自公开数据集 + 15GB 来自 CSTNET 被动采集）
    - Batch size = 32，训练步数 = 500,000
    - 学习率 = 2×10^{-5}，Warmup 比例 = 0.1
- 微调阶段：
    - Optimizer: AdamW
    - Epochs = 10
    - 学习率：Flow-level = 6×10^{-5}；Packet-level = 2×10^{-5}
    - Dropout = 0.5
- 环境：
    - 框架：PyTorch 1.8.0 + UER toolkit
    - 硬件：NVIDIA Tesla V100S GPU
- 伦理声明：
    - 所有流量采集均为被动的，且不包含任何个人识别信息；

- 已获 IRB 审查批准。

公开部分（约 15 GB）——可复现的数据

作者引用的两篇文献：

引用编号	数据集来源	内容说明	下载 / 获取途径
[9] Draper-Gil et al., 2016	**ISCX-VPN and Non-VPN Traffic Dataset**	来自加拿大网络安全研究所的加密流量（VPN / 非 VPN，6 种通信应用）	Canadian Institute for Cybersecurity (CIC)
[30] Sharafaldin et al., 2018	**CIC-IDS 2017 / CIC-IDS 2018 Dataset** 或 **CIC-DDoS2019**	包含 HTTPS、TLS、FTP、SSH 等多协议流量，可提取加密部分用于预训练	CIC Datasets Portal

这两份数据集的原始 PCAP 文件合计远超 15 GB；ET-BERT 的作者应该只提取了其中的加密流量段（如 **TLS、VPN、SSH** 等会话）来做预训练。

15GB 来自 CSTNET 被动采集——这一部分没找到

4.2 与最新方法对比 (Comparison with State-of-the-Art Methods)

作者将 ET-BERT 与 11 个经典/先进方法进行了系统对比：

对比方法分类：

1. 指纹匹配类：FlowPrint [35]
2. 统计特征类：AppScanner [34]、CUMUL [25]、BIND [1]、K-fp [11]

3. 深度学习类：DF [33]、FS-Net [20]、GraphDApp [31]、TSCRNN [19]、Deeppacket [23]
4. 预训练类：PERT [12]

Table 2: Comparison Results on Cross-Platform, ISCX-VPN-Service and ISCX-VPN-App datasets.

Dataset	Cross-Platform(iOS)				Cross-Platform(Android)				ISCX-VPN-Service				ISCX-VPN-App			
Method	AC	PR	RC	F1	AC	PR	RC	F1	AC	PR	RC	F1	AC	PR	RC	F1
AppScanner[34]	0.3205	0.2103	0.2173	0.2030	0.3868	0.2523	0.2594	0.2440	0.7182	0.7339	0.7225	0.7197	0.6266	0.4864	0.5198	0.4935
CUMUL[25]	0.2910	0.1917	0.2081	0.1875	0.3525	0.2221	0.2409	0.2189	0.5610	0.5883	0.5676	0.5668	0.5365	0.4129	0.4535	0.4236
BIND[1]	0.3770	0.2566	0.2715	0.2484	0.4728	0.3126	0.3253	0.3026	0.7534	0.7583	0.7488	0.7420	0.6767	0.5152	0.5153	0.4965
K-fp[11]	0.2155	0.2037	0.2069	0.2003	0.2248	0.2113	0.2104	0.2052	0.6430	0.6492	0.6417	0.6395	0.6070	0.5478	0.5430	0.5303
FlowPrint[35]	0.9254	0.9438	0.9254	0.9260	0.8698	0.9007	0.8698	0.8702	0.7962	0.8042	0.7812	0.7820	0.8767	0.6697	0.6651	0.6531
DF[33]	0.3106	0.2232	0.2179	0.2140	0.3862	0.2595	0.2620	0.2527	0.7154	0.7192	0.7104	0.7102	0.6116	0.5706	0.4752	0.4799
FS-Net[20]	0.3712	0.2845	0.2754	0.2655	0.4846	0.3544	0.3365	0.3343	0.7205	0.7502	0.7238	0.7131	0.6647	0.4819	0.4848	0.4737
GraphDApp[31]	0.3245	0.2450	0.2392	0.2297	0.4031	0.2842	0.2786	0.2703	0.5977	0.6045	0.6220	0.6036	0.6328	0.5900	0.5472	0.5558
TSCRNN[19]	-	-	-	-	-	-	-	-	-	0.9270	0.9260	0.9260	-	-	-	-
Deeppacket[23]	0.9204	0.8963	0.8872	0.9034	0.8805	0.8004	0.7567	0.8138	0.9329	0.9377	0.9306	0.9321	0.9758	0.9785	0.9745	0.9765
PERT[12]	0.9789	0.9621	0.9611	0.9584	0.9772	0.8628	0.8591	0.8550	0.9352	0.9400	0.9349	0.9368	0.8229	0.7092	0.7173	0.6992
ET-BERT(flow)	**0.9844**	0.9701	0.9632	0.9643	**0.9865**	0.9324	**0.9266**	**0.9246**	0.9729	0.9756	0.9731	0.9733	0.8519	0.7508	0.7294	0.7306
ET-BERT(packet)	0.9810	**0.9757**	**0.9772**	**0.9754**	0.9728	**0.9439**	0.9119	0.9206	**0.9890**	**0.9891**	**0.9890**	**0.9890**	**0.9962**	**0.9936**	**0.9938**	**0.9937**

Table 3: Comparison Results on ISCX-Tor, USTC-TFC and CSTNET-TLS 1.3 datasets.

Dataset	ISCX-Tor				USTC-TFC				CSTNET-TLS 1.3			
Method	AC	PR	RC	F1	AC	PR	RC	F1	AC	PR	RC	F1
AppScanner[34]	0.6722	0.3756	0.4422	0.3913	0.8954	0.8984	0.8968	0.8892	0.6662	0.6246	0.6327	0.6201
CUMUL[25]	0.6606	0.3850	0.4416	0.3918	0.5675	0.6171	0.5738	0.5513	0.5391	0.4942	0.5060	0.4904
BIND[1]	0.7185	0.4598	0.4515	0.4511	0.8457	0.8681	0.8382	0.8396	0.7964	0.7605	0.7650	0.7560
K-fp[11]	0.6472	0.5576	0.5849	0.5522	-	-	-	-	0.4036	0.3969	0.4044	0.3902
FlowPrint[35]	0.9092	0.3820	0.3661	0.3654	0.8146	0.6434	0.7002	0.6573	0.1261	0.1354	0.1272	0.1116
DF[33]	0.7533	0.6228	0.6010	0.5850	0.7787	0.7883	0.7819	0.7593	0.7936	0.7721	0.7573	0.7602
FS-Net[20]	0.6071	0.5080	0.5350	0.4590	0.8846	0.8846	0.8920	0.8840	0.8639	0.8404	0.8349	0.8322
GraphDApp[31]	0.6836	0.4864	0.4823	0.4488	0.8789	0.8226	0.8260	0.8234	0.7034	0.6464	0.6510	0.6440
TSCRNN[19]	-	0.9490	0.9480	0.9480	-	0.9870	0.9860	0.9870	-	-	-	-
Deeppacket[23]	0.7449	0.7549	0.7399	0.7473	0.9640	0.9650	0.9631	0.9641	0.8019	0.4315	0.2689	0.4022
PERT[12]	0.7682	0.4424	0.4446	0.4345	0.9909	0.9911	0.9910	0.9911	0.8915	0.8846	0.8719	0.8741
ET-BERT(flow)	0.8311	0.5564	0.6448	0.5886	**0.9929**	**0.9930**	**0.9930**	**0.9930**	0.9510	0.9460	0.9419	0.9426
ET-BERT(packet)	**0.9921**	**0.9923**	**0.9921**	**0.9921**	0.9915	0.9915	0.9916	0.9916	**0.9737**	**0.9742**	**0.9742**	**0.9741**

在论文 第3.4节 **Fine-tuning ET-BERT** 中，作者定义了两种微调策略：

类型	输入数据	模型目标	输出类别粒度
ET-BERT(packet)	单个 datagram（即一个或少数几个包）	捕获包级特征，细粒度分类	每个包的类别
ET-BERT(flow)	多个连续 datagram 组成的流（M=5 个包拼接）	捕获上下文依赖和时序特征	整个会话的类别

两者使用同一个预训练模型，只是在 **fine-tuning（微调）阶段输入不同**。

4.3 消融实验 (Ablation Study)

实验目的

评估各关键模块（预训练任务、BURST结构、微调策略等）的独立贡献。

数据集

- 使用 **ISCX-VPN-App** 数据集（17类应用，VPN流量）。

实验方法

设计6个对照模型，依次去除不同模块：

模型编号	去掉的组件	对应实验目的	微调方式	结果 (F1)
① 完整模型	无	作为基线	packet	93.95%
② w/o SBP	去掉Same-origin BURST预测	测试是否需BURST间关联	packet	89.98%
③ w/o MBM	去掉Masked BURST预测	测试是否需字节级上下文学习	packet	84.62%
④ w/o BURST	改用随机包输入	测试BURST结构有效性	packet	92.58%
⑤ flow微调	用整流输入	比较不同微调粒度	flow	73.87%
⑥ w/o Pretrain	直接监督训练	验证预训练必要性	packet	56.38%

结果与结论

- MBM任务贡献最大（↓9.33%）；
- SBP任务辅助性显著（↓3.97%）；
- BURST结构能捕获包间关系；
- 无预训练性能暴跌（↓37.6%）；

- 说明 **“BURST结构 + 双任务预训练”** 是性能关键来源。

4.4 可解释性分析 (Interpretability Analysis)

实验目的

分析 **为何加密流量仍可被模型区分**，探究其理论可行性。

数据集

- 使用全部任务中的主要数据集（ISCX-VPN、USTC-TFC、TLS 1.3 等），用于统计不同加密算法的比例与效果对比。

实验方法

(1) 随机性分析（Randomness Analysis）

- 工具：NIST 标准 15组随机性统计测试；
- 对象：5种常用加密算法（AES-GCM, AES-CBC, ChaCha20, ARC4, 3DES）；
- 指标：p-value（=1 表示完美随机）。

结果：

- 各算法均未达到完美随机性；
- 随机性缺陷（如比特分布差异）为模型提供可区分信号。

(2) 不同算法分布分析（Impact of Ciphers）

- 统计每数据集中加密算法的分布比例（见 *Figure 3*）；
- 比较模型在不同加密强度下的表现。

结果：

- 在使用弱随机性算法（ARC4, 3DES）的数据集上，ET-BERT 接近100%准确；
- 说明模型能识别出由加密实现差异导致的流量特征模式。

4.5 小样本学习分析 (Few-shot Analysis)

实验目的

验证 **ET-BERT 在样本稀缺场景下的鲁棒性与泛化能力**。

数据集

- **ISCX-VPN-Service**（12类VPN服务流量）

实验方法

1. 随机选取每类 500 条样本；
2. 仅使用 40%、20%、10% 数据进行训练；
3. 比较 ET-BERT 与传统模型（如 Deeppacket、FS-Net、BIND 等）的 F1 表现。

结果

训练数据比例	ET-BERT(PACKET) F1	对比
40%	95.78%	性能稳定
20%	98.33%	轻微波动
10%	91.55%	仍优于他法40%以上

结论：

- 传统监督学习方法性能大幅下降；
- ET-BERT 的预训练机制使其在低数据量场景中保持高效；
- 证明模型具备 **强迁移学习与样本高效性**。