Telegram Notifications

Module 6 of 7

Presenter: Vương Cường — Research Assistant, BAI Lab

What are Telegram Notifications?

Telegram Bot API = **Real-time notifications for long-running experiments**

Purpose

- **Training progress updates** - Monitor experiments remotely
- **Error notifications** - Immediate alerts when training fails
- **Completion alerts** - Know when experiments finish
- **Remote monitoring** - Stay informed without constant checking

Why Essential for Research?

- **Long-running experiments** - Training can take hours or days
- **Mobile accessibility** - Check progress from anywhere
- **Team notifications** - Shared experiment status updates
- **Immediate problem detection** - Fix issues quickly

Comparison with Traditional Monitoring

Method	Real-time	Mobile Access	Automation	Setup Complexity	Cost
Manual checking	No	No	No	None	Free
Email alerts	Delayed	Limited	Yes	Medium	Free
Slack notifications	Yes	Good	Yes	Medium	Free/Paid
SMS alerts	Yes	Full	Yes	High	Paid
Telegram	**Yes**	**Full**	**Yes**	**Low**	**Free**

Benefits Over Alternatives

- **Instant delivery** - Messages arrive within seconds
- **Rich formatting** - Markdown, emojis, and structured messages
- **Group notifications** - Team-wide experiment updates
- **File sharing** - Send plots, logs, and model checkpoints
- **Bot commands** - Interactive control of training processes

Key Advantage: Telegram provides immediate awareness of experiment status with minimal setup overhead.

Basic Notification Function

```python
# Simple notification function covers most needs
import requests
import os

def notify_telegram(message: str):
    """Send training updates to Telegram."""
    bot_token = os.environ["BOT_TOKEN"]
    chat_id = os.environ["CHAT_ID"]

    url = f"https://api.telegram.org/bot{bot_token}/sendMessage"
    data = {
        "chat_id": chat_id,
        "text": message,
        "parse_mode": "Markdown"
    }

    try:
        response = requests.post(url, data=data)
        response.raise_for_status()
    except requests.exceptions.RequestException as e:
        print(f"Failed to send Telegram notification: {e}")

# Usage examples
notify_telegram("🚀 Training started!")
notify_telegram(f"📊 Epoch {epoch}: loss={loss:.4f}, acc={acc:.3f}")
notify_telegram("✅ Training completed successfully!")
notify_telegram("❌ Training failed with error: {error_message}")
```

Advanced Features

```python
# Send images and files
def send_plot(plot_path: str, caption: str):
    """Send training plots to Telegram."""
    url = f"https://api.telegram.org/bot{os.environ['BOT_TOKEN']}/sendPhoto"

    with open(plot_path, 'rb') as photo:
        response = requests.post(url, data={
            "chat_id": os.environ["CHAT_ID"],
            "caption": caption
        }, files={"photo": photo})

# Send structured progress updates
def send_training_update(epoch, metrics):
    """Send formatted training progress."""
    message = f"""
🎯 **Training Update**
📈 Epoch: {epoch}
📊 Loss: {metrics['loss']:.4f}
🎯 Accuracy: {metrics['accuracy']:.3f}
⏱️ Time: {metrics['time']:.1f}s
🔥 GPU: {metrics['gpu_usage']:.1f}%
    """
    notify_telegram(message)
```

Step 1: Setup Telegram Bot

```
# 1. Create bot using @BotFather on Telegram
# - Open Telegram and search for @BotFather
# - Send /newbot command
# - Choose bot name and username
# - Copy the BOT_TOKEN

# 2. Get your Chat ID
# - Add your bot to a group or start private chat
# - Send a message to the bot
# - Visit: https://api.telegram.org/bot<BOT_TOKEN>/getUpdates
# - Find your chat_id in the response

# 3. Set environment variables
export BOT_TOKEN="1234567890:ABCdefGHIjklMNOpqrsTUVwxyz"
export CHAT_ID="987654321"

# Or add to .env file (don't commit to git!)
echo "BOT_TOKEN=your_bot_token_here" >> .env
echo "CHAT_ID=your_chat_id_here" >> .env
```

Step 2: Add Notifications to Training Script

```python
# Add to src/training/train.py
import requests
import os
from datetime import datetime

def notify_telegram(message: str):
    """Send message to Telegram."""
    bot_token = os.environ.get("BOT_TOKEN")
    chat_id = os.environ.get("CHAT_ID")

    if not bot_token or not chat_id:
        print("Telegram credentials not found, skipping notification")
        return

    url = f"https://api.telegram.org/bot{bot_token}/sendMessage"
    requests.post(url, data={
        "chat_id": chat_id,
        "text": message,
        "parse_mode": "Markdown"
    })
```

Step 2: Add Notifications to Training Script (contd.)

```python
def train_with_notifications():
    """Training with Telegram notifications."""
    try:
        # Start notification
        notify_telegram(f"🚀 Training started at {datetime.now().strftime('%H:%M:%S')}")

        for epoch in range(10):
            # Simulate training
            train_loss = 1.0 - (epoch * 0.1)
            val_acc = 0.5 + (epoch * 0.05)

            # Periodic updates (every 2 epochs)
            if epoch % 2 == 0:
                notify_telegram(
                    f"📊 **Epoch {epoch}**\n"
                    f"Loss: {train_loss:.3f}\n"
                    f"Accuracy: {val_acc:.3f}"
                )

            print(f"Epoch {epoch}: loss={train_loss:.3f}, acc={val_acc:.3f}")

        # Completion notification
        notify_telegram("✅ Training completed successfully!")

    except Exception as e:
        # Error notification
        notify_telegram(f"❌ Training failed: {str(e)}")
        raise

if __name__ == "__main__":
    train_with_notifications()
```

Step 3: Integration with W&B

```python
# Combined W&B + Telegram notifications
import wandb

def train_with_full_tracking():
    """Training with both W&B and Telegram."""
    wandb.init(project="bailab-base-research")
    notify_telegram("🚀 Starting experiment: " + wandb.run.name)

    for epoch in range(wandb.config.epochs):
        # Training code...

        # Log to W&B
        wandb.log({"train/loss": train_loss, "val/accuracy": val_acc})

        # Telegram updates every 10 epochs
        if epoch % 10 == 0:
            notify_telegram(
                f"📈 [{wandb.run.name}] Epoch {epoch}\n"
                f"🎯 Accuracy: {val_acc:.3f}\n"
                f"📊 View: {wandb.run.url}"
            )

    # Final notification with W&B link
    notify_telegram(f"✅ Experiment complete!\n📊 Results: {wandb.run.url}")
```

Security and Best Practices

Token Management

```
# NEVER commit tokens to git
# Use environment variables
import os
from dotenv import load_dotenv

load_dotenv()  # Load from .env file
BOT_TOKEN = os.environ.get("BOT_TOKEN")
CHAT_ID = os.environ.get("CHAT_ID")

# Add to .gitignore
echo ".env" >> .gitignore
```

Rate Limiting and Error Handling

```python
import time
from functools import wraps

def rate_limited(max_per_minute=30):
    """Decorator to rate limit Telegram messages."""
    def decorator(func):
        calls = []

        @wraps(func)
        def wrapper(*args, **kwargs):
            now = time.time()
            calls[:] = [call for call in calls if call > now - 60]

            if len(calls) >= max_per_minute:
                print("Rate limit exceeded, skipping notification")
                return

            calls.append(now)
            return func(*args, **kwargs)
        return wrapper
    return decorator

@rate_limited(max_per_minute=20)
def notify_telegram(message: str):
    # Implementation here...
    pass
```

Notification Strategy

When to Send Notifications

```
# Strategic notification points
notify_telegram("🚀 Training started")          # Always
notify_telegram("📊 Epoch progress")            # Every 10-20 epochs
notify_telegram("💾 Checkpoint saved")          # Major milestones
notify_telegram("✅ Training completed")        # Always
notify_telegram("❌ Error occurred")            # Always
notify_telegram("⚠️ Early stopping triggered")  # Important events
```

Message Formatting Best Practices

```python
# Use emojis for quick visual recognition
# Structure information clearly
# Include relevant links (W&B, logs)
# Keep messages concise but informative

def format_progress_message(epoch, total_epochs, metrics):
    """Format training progress message."""
    progress_bar = "█" * int(10 * epoch / total_epochs) + "░" * (10 - int(10 * epoch / total_epochs))

    return f"""
🎯 **Training Progress**
{progress_bar} {epoch}/{total_epochs} ({100*epoch/total_epochs:.1f}%)

📊 **Metrics**
• Loss: {metrics['loss']:.4f}
• Accuracy: {metrics['accuracy']:.3f}
• LR: {metrics['lr']:.2e}

⏱ ETA: {metrics.get('eta', 'Unknown')}
    """
```

Team Collaboration

Group Notifications

```
# Create dedicated research group
# Add all team members and bots
# Use structured messages for clarity
# Consider separate channels for different projects

def send_team_update(project_name, researcher, status, details):
    """Send structured team update."""
    message = f"""
🔬 **{project_name}**
👤 Researcher: {researcher}
📋 Status: {status}
📝 Details: {details}
🔗 Dashboard: [View Results](https://wandb.ai/...)
    """
    notify_telegram(message)
```

What We Covered

✅ **Telegram fundamentals** - Real-time experiment monitoring

✅ **Advantages over traditional methods** - Mobile access and automation

✅ **Essential implementation** - Bot setup and notification functions

✅ **Lab-specific integration** - Training script integration

✅ **Best practices** - Security, rate limiting, and team collaboration

Key Takeaways

1. **Telegram provides immediate experiment awareness** - Never miss important updates
2. **Simple setup with powerful features** - Rich formatting and file sharing
3. **Team collaboration enablement** - Shared experiment status
4. **Mobile accessibility** - Monitor experiments from anywhere
5. **Integration with other tools** - Works perfectly with W&B and other platforms

Impact on Research Workflow

- **Reduced manual monitoring** - Automatic updates instead of constant checking
- **Faster problem resolution** - Immediate error notifications
- **Better team coordination** - Shared visibility into experiment progress
- **Mobile research management** - Monitor experiments on the go

Next Steps

➡️ **Module 7: Integrated Workflow Demo** - Putting it all together
➡️ Setup Telegram bot for your current research project
➡️ Configure team notification channels